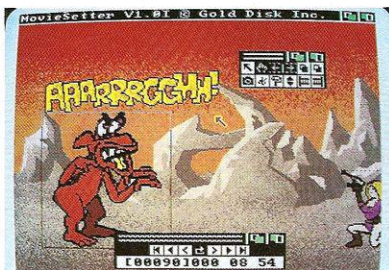
Anno XI - N. 81 - 25 Gennaio 25 Febbraio1991 - Sped. Abb. Post. Gr III/70 - CR - Distr.: Parrini

AMIGA 500

Disco esterno da 5"25

COMPILATORI

- Tutti i "C" di Amiga
- Turbo Pascal
- MS-DOS

MOVIE SETTER

LA FABBRICA DEI CARTOON ORA IN ITALIANO

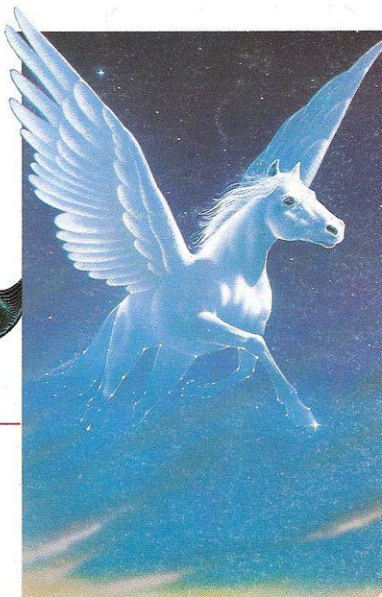
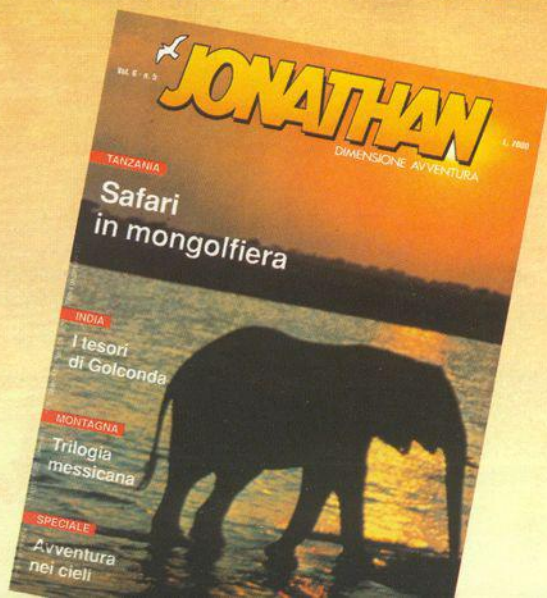
STAMPANTI

- Come sceglierla
- Hard copy per Amiga e C64

Grafici di funzione tridimensionali

Volare verso l'assembly MS-DOS

GEOS, Viaggio nelle icone ● MS-DOS, primi comandi ● Videogames per Amiga ● Modem per A2000 ● C1 Text, ultima versione ● Amiga Dos ● MS-DOS, Microcad in C

JONATHAN
DIMENSIONE AVVENTURA
TANZANIA
Safari
in mongolfiera
INDIA
I tesori
di Golconda
MONTAGNA
Trilogia
messicana
SPECIALE
Avventura
nei cieli

Sommario

Amigames

Sono soltanto tre i videogames recensiti questo mese; ciò per venire incontro alle numerose richieste dei nostri lettori, che preferiscono un maggior numero di pagine dedicate ad articoli tecnici.

Spazio 64 / 128

Amiga

Amiga + Ms - Dos

Mondo Dos

COMMODORE COMPUTER CLUB

Direttore: Alessandro de Simone
Coordinatore: Marco Miotti

Redazione / Collaboratori:
Davide Ardizzone - Claudio Baiocchi
Luigi Callegari - Umbero Colapicchioni
Donato De Luca - Carlo D'Ippolito
Valerio Ferri - Michele Maggi
Giancarlo Mariani - Domenico Pavone
Armando Sforzi - Dario Pistella
Fabio Sorgato- Valentino Spataro
Franco Rodella- Stefano Simonelli
Luca Viola

Direzione:
Via Mosè, 22 cap. 20090 OPERA (Mi)

Telefono 02 / 57.60.63.10
Fax 02 / 57.60.30.39
BBS 02 / 57.60.52.11

Pubblicità:
Leandro Nencioni (dir. vendite)
Via Mose', 22 20090 Opera (Mi)
tel. 02 / 57.60.63.10

Emilia Romagna:
Spazio E
P.zza Roosvelt, 4 cap. 40123 Bologna
Tel. 051 / 23.69.79

Toscana, Marche, Umbria
Mercurio s.r.l. Via Rodari, 9
S. G.nni Valdarno (Ar)
Tel. 055 / 94.74.44

Lazio, Campania
Spazio Nuovo
Via P. Foscari, 70
cap. 00139 Roma
tel. 06 / 81.09.679

Abbonamenti: Liliana Spina
Arretrati e s/w: Lucia Dominoni

Tariffe: Prezzo per copia L. 6000
Abbonamento annuo (11 fascicoli) L. 60000
Estero: L. 100000 - Indirizzare versamenti a:
Systems Editoriale Srl c/c 37952207 oppure
inviare comune assegno bancario non
trasferibile e barrato due volte a:
Systems Editoriale Srl (servizio arretrati)
Via Mose', 22
cap. 20090 OPERA (Mi)

Composizione: Systems Editoriale
La Litografica Srl Busto Arsizio (Va)

Registrazioni: Tribunale di Milano
n. 370 del 2/10/82

Direttore Responsabile: Michele Di Pisa

Spedizioni in abbonamento postale gruppo
III. Pubblicita' inferiore al 70%

Distributore: Parrini - Milano

Periodici Systems: Banca Oggi -
Commodore Club (disco) - Commodore
Computer Club - Commodore Computer
Club (disco, produzione tedesca) - Computer
- Computer disco - Electronic Mass Media
Age - Energy Manager - Hospital
Management - Jonathan - Nursing '90 - PC
Programm (disco) - Personal Computer -
Security - Software Club (cassetta ed.
italiana) - TuttoGatto - Videoteca
VR Videoregistrare

NEW DEAL

Siamo così giunti al gennaio del 1991. Data che, per alcuni versi, rappresenta una svolta fondamentale; da un altro punto di vista, invece, si potrebbe affermare che non c'è alcun cambiamento in atto.

Nel N. 1 di Commodore Computer Club, infatti, si parlava esclusivamente dei computer Commodore serie Pet; dopo pochi numeri il Vic 20, novità del momento, occupò la maggior parte delle pagine. Ma era già tempo di cambiamenti: del Commodore 64 si iniziò a parlare diffusamente fin dal n. 5, e se ne parla ancora oggi, pur se in tono nettamente minore.

Lo spazio del fascicolo che state ora leggendo, però, è oggi ripartito, come allora, in proporzione diretta con la diffusione dei computer attualmente in voga, ma con una novità, ripetto ai tempi antichi, consentita proprio dall'evoluzione dei computer moderni.

Ci riferiamo alla compatibilità, o meglio, alla versatilità offerta dai nuovi sistemi computerizzati che permettono uno scambio di dati, se non addirittura di programmi, tra macchine dall'architettura fondamentalmente diversa tra loro.

Tra **Amiga** ed il mondo **Ms - Dos**, ad esempio, è possibile uno **scambio di dati** tra i due "ambienti".

Chi, infatti, possiede entrambi i computer (ed un banalissimo cavetto in grado di collegarli via interfaccia Rs - 232), può trasferire in modo agevole files di testo, immagini Iff e programmi (scritti in C oppure in AmigaBasic o Quick Basic, a seconda dei casi); semplici(!) adattamenti consentono di trattare i files trasferiti nel nuovo sistema "ospite".

Chi, invece, dispone di un solo computer, non ha che l'imbarazzo della scelta per effettuare scambi di dati. Con l'Amiga, come infatti è noto, è sufficiente il programma **DosToDos** per riversare files di formato Amiga su dischetti formattati in ambiente Ms - Dos. Chi, poi, vuole un'emulazione più radicale, deve procurarsi un'economica scheda, da inserire nella pancia di Amiga, per avere a disposizione un vero e proprio computer Ms - Dos. Il viceversa (trasformare un Ms - Dos in Amiga) non è ancora possibile, e dubitiamo che ciò possa avvenire; a meno che i progettisti della Commodore, realizzando una scheda adeguata da inserire in un PC compatibile, dimostrino una fantasia commerciale finora tenuta abilmente nascosta.

Eufemismi a parte, il fenomeno Amiga, nato con ottimi auspici (a dispetto delle notevoli pecche iniziali), si è gradualmente affievolito fino a diventare, purtroppo, quasi esclusivamente sinonimo di videogame. E ciò potrebbe anche andar bene, se il campo dei videogame non fosse già appannaggio quasi esclusivo di altri colossi orientali. E' mai possibile che la Commodore non abbia pensato alla produzione di una scheda Amiga da inserire in un PC? Eppure, in precedenza, ha dimostrato di avere una fantasia piuttosto spregiudicata nel proporre computers (C/16, Plus 4, C/128) dal futuro certamente meno roseo di una scheda Amiga. Per fortuna le software house stanno provvedendo a colmare il divario ancora esistente tra il mondo Ms - Dos e quello Amiga, producendo programmi che nulla hanno da invidiare all'Amiga; e ci riferiamo, ovviamente, al campo del divertimento, dal momento che il campo professionale, piaccia o no, appartiene al monopolio Ms - Dos.

Alessandro de Simone

LA VOSTRA POSTA

(a cura di A. de Simone)

Mouse dispettoso

Al mio computer accade una cosa stranissima: quando lavoro nelle belle giornate di sole, il mouse non risponde in modo adeguato e si comporta come se, di tanto in tanto, la sferetta si bloccasse. E' probabile che il funzionamento del mouse sia legato, in qualche modo, all'umidità o al clima troppo secco dell'ambiente?

(Francesco A. Torino)

No, a meno che non lavori abitualmente nella vasca da bagno mentre ti fai la doccia (e in questo caso, attento alle scosse elettriche!).

Scherzi a parte, è molto probabile che il motivo sia dovuto esclusivamente alla "trasparenza" della custodia del mouse. Per meglio chiarire il concetto, è bene soffermarsi sul modo in cui un qualsiasi mouse funziona.

Quando spostiamo l'accessorio sul tavolo (o meglio, sul tappetino specifico), la sferetta trasmette il movimento di rotazione a due cilindretti che, a loro volta, sono solidali ad altrettanti dischetti forati. Da una parte e dall'altra di ciascun dischetto sono presenti, rispettivamente, un **led** ed un **sensore** alla luce prodotta dal led. Quando, grazie alla rotazione della sferetta, i dischetti vengono posti in rotazione, il fascio luminoso prodotto dal led viene interrotto ed il sensore, grazie ad una particolare circuiteria elettronica, è in grado di stabilire il movimento impresso al mouse. La combinazione delle due rotazioni, ed il programma che sovrintende al mouse, sono quindi in grado di interpretare correttamente lo spostamento generato.

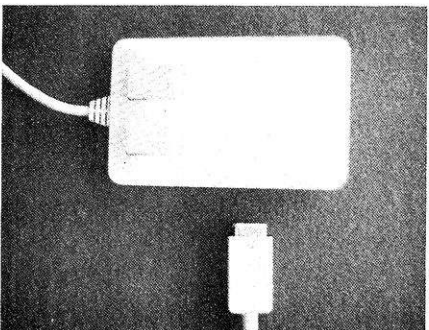
Il tutto, ovviamente, avviene in modo rapidissimo e, soprattutto, senza che l'utente se ne accorga.

Per un corretto funzionamento del mouse, dunque, è indispensabile che si verifichi l'interruzione del fascio luminoso. Se, pertanto, il sensore luminoso non è in grado di avvertire tale interruzione, "riterrà" che il mouse non si è spostato, almeno nella direzione in cui non riesce ad individuare interruzioni di luce.

Gli involucri di alcuni mouse sono realizzati in materiale plastico molto sottile e, per giunta, di colore bianco (ottimo diffusore di luce). Se un mouse di questo tipo viene investito da una quantità rilevante di luce (luce diretta del sole, lampada da tavolo molto vicina all'area di azione del mouse, ecc.), una parte di questa riesce a penetrare al suo interno ed a colpire uno (o entrambi) i sensori luminosi, falsandone la lettura.

Per verificare se è questo il motivo del malfunzionamento, prova a tirar giù le tapparelle quando il fastidioso fenomeno si verifica. Se questo scompare, il rimedio consiste nel verniciare di nero l'involucro; e comunque, senza arrivare a questi estremi, a fare in modo che la quantità di luce che giunge sul mouse sia la minima possibile.

L.m. difficile

Sul n. 75 avevate consigliato, ad un lettore, l'acquisto del volume "Guida di riferimento del programmatore" per imparare il linguaggio macchina del C/64. Nonostante io abbia seguito il vostro consiglio, non capisco

Insegnante A

Sono un insegnante di informatica nelle scuole elementari e rispondo volentieri alla richiesta di informazioni lanciata attraverso le vostre pagine.

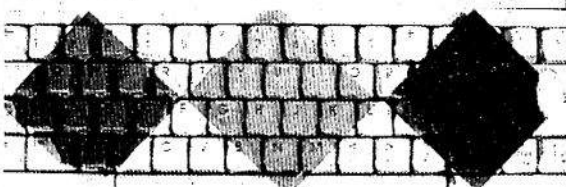
Lavoro presso la direzione didattica di Civitavecchia II. Il circolo didattico comprende tre plessi scolastici, in ognuno dei quali è attiva un'aula di informatica. Ciascuna di queste comprende otto sistemi completi, ognuno formato da: C/64; plotter 1520; drive 1571; monitor 1901; stampante Mps-803.

Dei 900 alunni, usufruiscono delle aule di informatica ben 520 ragazzi (escludendo, per ovvi motivi, le sole classi prime) pur se, come insegnanti di informatica, siamo soltanto in due (l'anno scorso ce ne era uno solo!). Le aule lavorano a pieno ritmo anche grazie alla partecipazione di insegnati appartenenti all'area logico-matematica ed i risultati conseguiti sono incoraggianti, pur se è giocoforza sistemare due / tre allievi per ciascun sistema computerizzato.

(Alberto Di Felice)

Insegnate B

Non sono d'accordo sulla volontà di additare i "cattivi" delle scuole italiane pubblicando i nomi delle scuole in cui, nonostante la presenza di aule di informatica, non si svolgono lezioni specifiche a causa di una rugginosa buracrazia. Penso, invece, che sia necessario sforzarsi per coinvolgere maggiormente insegnanti, presidi e personale di segreteria poco sensibili alle esigenze della didattica.

(Antonio B.)

Le due lettere vengono riportate un po' condensate per esigenze di spazio, e di questo fatto mi scuso con gli autori delle stesse.

Come è possibile notare, comunque, non c'è assolutamente l'intenzione di riportare **solo** i nomi dei cattivi. La pubblicazione della prima lettera lo dimostra.

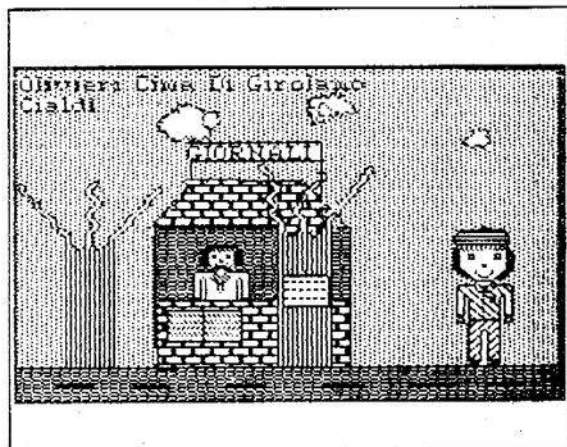
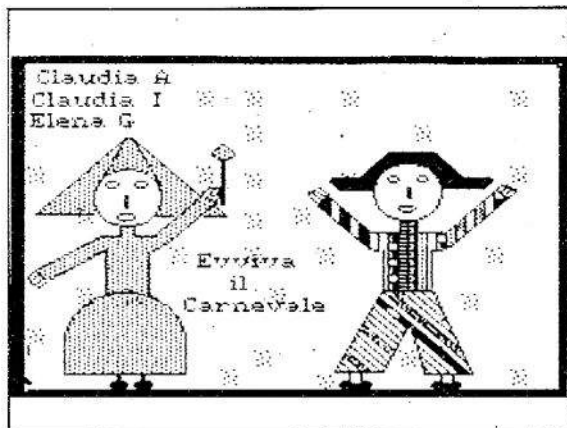
Io penso, però, che una certa forza di persuasione può anche venire dalla "base".

Immaginiamo di metterci nei panni di un nostro lettore, studente delle medie superiori, il quale legge che in una scuola elementare alcuni insegnati fanno a pugni (si fa per dire, mi auguro...) per accedere all'aula di informatica. Certamente gli verranno in mente le motivazioni addotte dal proprio preside (o insegnante che sia) i quali, con una scusa o con un'altra, rifiutano l'accesso al laboratorio (pagato, non dimentichiamolo, con i soldi di noi contribuenti).

Ora pensiamo ad un altro insegnate che, nel laboratorio, si limita a fare il minimo indispensabile (a causa della propria disinformazione o mancanza di aggiornamento professionale) e che non può impedire di far manifestare l'esperienza di uno studente più informato di lui.

Se questo insegnante ha un po' di dignità, i casi sono due: o **corre** ad aggiornarsi, oppure rischia costantemente magre figure con i propri allievi (che, giustamente, avranno più fiducia nel loro compagno che nel docente).

Facile, poi, immaginare i pensieri che attraversano la mente dello studente di una scuola per programmatori (costretto a studiare informatica in un'aula, magari, dotata di un solo computer) quando viene a sapere che in una scuola elementare sono a disposizione ben otto macchine.

Nascondere alcune situazioni particolari o, peggio, avallare il comportamento di una certa parte del corpo docente (pur se minima) che si disinteressa dell'aggiornamento professionale, non fa altro che peggiorare la situazione; confortando, paradossalmente, coloro che vogliono insistere nel trascurare l'introduzione dell'informatica nelle scuole.

Libri per C/64

Il volume **"600 consigli e trucchi per Commodore"** è un libro di ben 360 pagine edito dalla casa editrice **Apogeo** al prezzo di **L. 48000**. Scritto da **Lou Sander**, suddiviso in 35 capitoli, contiene centinaia di trucchi e consigli "estratti" dalla pubblicazione americana Commodore Magazine (rubrica **Tips & Tricks**) ed opportunamente tradotti in italiano.

Alla notevole raccolta appartengono anche trucchi e consigli per gli utenti del **Plus 4** e del **C/128** (perfino in modo **CP/M**!) che potranno, così, rasserenarsi per non essere stati del tutto abbandonati dalle case editrici specializzate.

Non mancano, ovviamente, suggerimenti anche per le periferiche dei popolari computers: drives, stampanti, mouse, joy eccetera.

Un libro da leggere e, soprattutto, da sperimentare trucco per trucco, pagina dopo pagina.

quasi nulla di quanto c'è scritto su quel libro (ad esempio, che cos'è il bit, e a che serve spostare a sinistra i bit di un byte?). Ora mi domando: vale la pena acquistare anche il vostro fascicolo "Speciale linguaggio macchina" oppure rischio di continuare a non capire nulla?
(Marco R. - Roma)

Di recente ho fatto cambiare le candele alla macchina e, per caso fortuito, mi è capitato di annotare il tempo impiegato dal meccanico a compiere l'operazione.

Tra una cosa e l'altra (sposta la macchina, apri il cofano, svita le candele, prendi nota del codice richiesto, vai in magazzino, ritorna, avvita, prova a mettere in moto e incassa il denaro) se ne è andato quasi un quarto d'ora. Subito dopo, a casa di amici, mi è capitato di assistere ad un cartone animato televisivo in cui, mentre un mostro meccanico avanza a grandi passi seminando morte, terrore e/o distruzione, alla periferia della città visitata dal mostro alcuni eroi **stanno riparando** le armi micidiali di un'astronave. Badate bene: non stanno cambiando le candele, stanno effettuando una riparazione che, a giudicare dalle sequenze, sembra richiedere quasi una fabbricazione ex-novo dell'arma stessa.

Tuttavia il tempo richiesto dalla riparazione (solo pochi secondi!) coincide, guarda caso, con l'arrivo del mostro che, ovviamente, viene subito distrutto per mezzo di lame rotanti, missili esplodenti e sfere sputacchianti.

L'episodio, purtroppo, si lega benissimo con quello che molti giovani sono abituati a ritenere, a causa della persuasione occulta regalataci da anni di spettacoli televisivi fuorvianti. Chi, abituato ai cartoni di infima fattura, vede in azione un computer, ritiene che basta premere un paio di tasti per scatenare guerre mondiali, far spostare pianeti dalle loro orbite o altre azioni similmente altruistiche (i cattivi, si sa, sono sempre gli altri).

Per "capire" un computer, ahimè, è necessaria tanta, tanta pazienza e, soprattutto, la volontà di dedicare *centinaia* di ore (senza esagerazioni) per impadronirsi delle varie tecniche di programmazione.

Chi non può (o non vuole) dedicare tanto tempo, può limitarsi al Basic; chi, poi, vuol dedicare tempo ancora minore, si accontenti di usare programmi già pronti pe l'uso.

Non c'è proprio nulla di male nel rinunciare a scoprire tecniche di programmazione avanzate o interazioni con il cuore della macchina.

❑❒

1230 compatibile

La stampante Commodore Mps 1230 è compatibile con il Geos 1.2 e con il programma di Word Processor Speed Script?
(Paolo Cecchi - Firenze)

La stampante Mps-1230 possiede, sul retro, due connettori per il collegamento con il computer.

Il primo ha la nota forma rotonda (tipo drive del C/64, per intenderci) per collegarla, appunto, ad un qualsiasi computer della vecchia generazione (Vic 20, C/64, C/16, Plus 4, C/128); l'altro, invece, è un connettore standard di tipo Centronics che consente di collegare la stampante a computer moderni (Ms-Dos e compatibili vari, Amiga e così via).

Per ciò che riguarda la compatibilità con Geos, la Commodore assicura che c'è. Numerosi lettori, inoltre, lo confermano inviandoci lettere scritte, appunto, con C/64, Geos e Mps-1230.

❑❒

W/P; meglio Amiga?

Posseggo un C/128 e tra breve lo sostituirò con un Amiga. Sarà possibile trovare un w/p che sia in grado di non farmi rimpiangere il mio Geowrite 128 (avevo pensato a C1-Text, che ne dite?).
(Gianluca Marangoni - Cotignola)

Ueilà, ma scherziamo? Un **qualsiasi** w/p moderno per sistemi superiori (Amiga ed Ms-Dos) è certamente migliore del sistema C/128 + Geos. Il progresso non è un bluff.

C1 Text, come hai modo di informarti in altra parte della rivista, è certamente un ottimo w/p, da acquistare (ovviamente) in confezione **originale** grazie alla presenza del voluminoso e chiaro manuale d'uso in lingua italiana.

Per quanto riguarda la scelta del modello di Amiga da acquistare (500 oppure 2000?) non possiamo che ripetere quanto detto altre volte: elenca su un foglio vantaggi e svantaggi offerti da ciascun modello; poi tira le somme e decidi.

Concludo sottolineando che mi fa piacere, come affermi, che ad **Argnani** (P.zza libertà 5/a) in provincia di **Ravenna**, sia presente un Commodore Point dove, oltre ad una certa competenza e professionalità, hai trovato prezzi molto bassi.

❑❒

Mai pubblicato

Nel vostro programma "Gestione albergo" (pubblicato sul numero di luglio agosto 1985) mancano alcune linee di programma. Potete pubblicarle?
(V. Speranza)

Sul numero citato non compare il programma citato dal nostro lettore; nè, in altre occasioni, è mai stato pubblicato un programma del genere.

E' probabile che sia stata fatta confusione con il nome delle riviste in possesso dal sig. Speranza.

SYSTEMS EDITORIALE PER TE

La voce
Aggiunge al C/64 nuovi comandi Basic che consentono sia di far parlare il computer, sia di farlo Cantare! Diversi esempi allegati.
Cassetta: L. 12000 - Disco: L. 15000

Raffaello
Un programma completo per disegnare, a colori, con il C/64: linee, cerchi, quadrati, eccetera. Valido sia per disegno a mano libera che geometrico.
Cassetta: L. 10000

Oroscopo
Devi solo digitare la data di nascita e le coordinate geografiche del luogo che ti ha dato i natali. Vengono quindi elaborate le varie informazioni (case, influenze dei segni astrali, eccetera) e visualizzato un profilo del tuo carattere. Valido per qualsiasi anno, è indicato sia agli esperti sia ai meno introdotti. E' allegata una tabella delle coordinate delle più note città italiane e l'elenco delle ore legali in Italia dal 1916 al 1978.
Cassetta: L. 12000 - Disco: L. 12000

Computer Music
Cassetta contenente numerosi brani di successo da far eseguire, in interrupt, al tuo C/64 sfruttando, fino in fondo, il suo generatore sonoro (SID).
Cassetta: L. 12000

Gestione Familiare
Il più noto ed economico programma per controllare le spese e i guadagni di una famiglia.
Cassetta: L. 10000 - Disco: L. 10000

Banca Dati
Il più noto ed economico programma per gestire dati di qualsiasi natura.
Cassetta: L. 10000 - Disco: L. 10000

Matematica finanziaria
Un programma completo per la soluzione dei più frequenti problemi del settore.
Cassetta: L. 10000 - Disco: L. 20000

Analisi di bilancio
Uno strumento efficace per determinare con precisione i calcoli necessari ad un corretto bilancio.
Cassetta: L. 10000 - Disco: L. 20000

Corso di Basic
Confezione contenente quattro cassette per imparare velocemente le caratteristiche delle istruzioni Basic del C/64 e i rudimenti di programmazione. Interattivo.
Cassetta: L. 19000

Corso di Assembler
Un corso completo su cassetta per chi ha deciso di abbandonare il Basic del C/64 per addentrarsi nello studio delle potenzialità del microprocessore 6502. Interattivo.
Cassetta: L. 10000

Logo Systems
Il linguaggio più facile ed intuitivo esistente nel campo dell'informatica; ideale per far avvicinare i bambini al calcolatore.
Diversi esempi allegati.
Cassetta: L. 6500

Compilatore Grafico Matematico
Uno straordinario programma compilatore, di uso semplicissimo, che permette di tracciare, sul C/64, grafici matematici Hi-Res ad altissima velocità. Esempi d'uso allegati.
Cassetta: L. 8000

Emulatore Ms-Dos e Gw-Basic
Un prodotto, unico nel suo genere, che permette di usare, sul C/64 dotato di drive, la sintassi tipica del più diffuso sistema operativo del mondo. Ideale per studenti.
Solo su disco: L. 20000

Emulatore Turbo Pascal 64
Permette di usare le più importanti forme sintattiche del linguaggio Turbo Pascal (anche grafiche!) usando un semplice C/64 dotato di drive.
Ideale per studenti.
Disco: L. 19000

Speciale drive
Questo speciale fascicolo costituisce una guida di riferimento per le unità a disco del C64/128.
Comprende anche un velocissimo turbo-disk più la mappa completa della memoria del drive.
Fascicolo + disco: L. 12000

Utility 1
Un dischetto pieno zeppo di programmi speciali per chi opera frequentemente con il drive.
Disco: L. 12000

Utility 2
Seconda raccolta di utility indispensabili per realizzare sofisticate procedure di programmazione.
Disco: L. 15000

Graphic Expander 128
Per usare il C/128 (in modo 128 e su 80 colonne) in modo grafico Hi-res. Aggiunge nuove, potenti istruzioni Basic per disegnare in Hi-Res con la massima velocità in modalità 80 colonne.
Disco: L. 27000

Directory
Come è noto, a partire dal N. 10 di "Software Club" (la rivista su disco per l'utente dei "piccoli" computer Commodore), vengono riportati tutti i listati, in formato C/64-C/128, pubblicati su "Commodore Computer Club".
In precedenza tali listati venivano inseriti, mensilmente, in un dischetto, di nome "Directory", che oltre ai programmi di C.C.C. ospitava decine di altri file tra cui musiche nell'interrupt, giochi, listati inviati dai lettori e altro.
Ogni disco, dal prezzo irrisorio, contiene quindi una vera miniera di software. Ordinando i dischetti di "Directory" si tenga conto che al N. 1 corrispondeva il contenuto del N. 34 di "Commodore Computer Club", al N. 2 il N. 35 e così via.
Ogni dischetto: L. 10000

Super Tot '64
La nuova e completa edizione del programma Tot 13 con tutti i sistemi di riduzione e di condizionamento.
Ampia sezione dedicata alla teoria.
fascicolo + disco: L. 15000

Amiga Totospeed
Finalmente anche per Amiga un programma orientato alla compilazione delle schedine totocalcio.
Fai tredici conil tuo Amiga.
disco: L. 20000

SYSTEMS EDITORIALE PER TE

Disk'o'teca
Grazie a questa nutrita raccolta di brani musicali potrete divertirvi ascoltando i migliori brani prodotti dai vostri beniamini, oltre a una serie di composizioni prodotte "in casa".
In omaggio un bellissimo poster di Sting.

Disco: L.15.000

Assaggio di primavera
Esclusivo!
In un'unica confezione potrete trovare ben due cassette di videogiochi assieme a un comodo e funzionale joystick.

Cassette: L. 15.000

LIBRI TASCABILI

64 programmi per il C/64
Raccolta di programmi (giochi e utilità) semplici da digitare e da usare. Ideale per i principianti. **(126 pag.)**

L. 4800

I miei amici C/16 e Plus/4
Il volumetto, di facile apprendimento, rappresenta un vero e proprio mini-corso di Basic per i due computer Commodore. Numerosi programmi, di immediata digitazione, completano la parte teorica. **(127 pag.)**

L. 7000

62 programmi per C/16, Plus/4
Raccolta di numerosi programmi, molto brevi e semplici da digitare, per conoscere più a fondo il proprio elaboratore.
Ideale per i principianti. **(127 pag.)**

L. 6500

Micro Pascal 64
Descrizione accurata della sintassi usata dal linguaggio Pascal "classico". Completa il volume un programma di emulazione del PL/O sia in formato Microsoft sia in versione C/64 (da chiedere, a parte, su disco). **(125 pag.)**

L. 7000

Dal registratore al Drive
Esame accurato delle istruzioni relative alle due più popolari periferiche del C/64. Diversi programmi applicativi ed esempi d'uso. (94 pag.)

L. 7000

Il linguaggio Pascal
Esame approfondito della sintassi usata nel famoso compilatore. (112 pag.)

L. 5000

Simulazioni e test per la didattica
Raccolta di numerosi programmi che approfondiscono e tendono a completare la trattazione già affrontata sul precedente volume. **(127 pag.)**

L. 7000

Dizionario dell'Informatica
Dizionario inglese-italiano di tutti i termini usati nell'informatica. (Edizione completa). **(385 pag.)**

L. 10000

Word processing: istruzioni per l'uso
Raccolta delle principali istruzioni dei più diffusi programmi di w/p per i sistemi Ms-Dos: Word-Star, Samna, Multimate Advantage, Word 3. **(79 pag.)**

L. 5000

Unix
Un volumetto per saperne di più sul sistema operativo professionale per eccellenza.
Un necessario compendio per l'utente sia avanzato che inesperto **(91 pag.)**

L. 5000

ABBONAMENTO

Commodore Computer Club
11 fascicoli: L. 60.000

ARRETRATI

Ciascun numero arretrato di C.C.C. L. 6.000

Come richiedere i prodotti Systems

Coloro che desiderano procurarsi i prodotti della Systems Editoriale devono inviare, oltre alla cifra risultante dalla somma dei singoli prodotti, L. 3500 per spese di imballo e spedizione, oppure L. 6000 se si desidera la spedizione per mezzo raccomandata.
Le spese di imballo e spedizione sono a carico della Systems se ciascun ordine è pari ad almeno L. 50000.
Per gli ordini, compilare un normale modulo di C/C postale indirizzato a:

C/C Postale N. 37 95 22 07
Systems Editoriale Srl
Via Mosè, 22
20090 Opera (MI)

Non dimenticate di indicare chiaramente, sul retro del modulo (nello spazio indicato con "Causale del versamento"), non solo il vostro nominativo completo di recapito telefonico, ma anche i prodotti desiderati ed il tipo di spedizione da effettuare.
Per sveltire la procedura di spedizione sarebbe opportuno inviare, a parte, una lettera riassuntiva dell'ordine effettuato, allegando una fotocopia della ricevuta del versamento.
Chi volesse ricevere più celermente la confezione deve inviare la somma richiesta mediante assegno circolare, oppure normale assegno bancario (non trasferibile o barrato due volte) intestato a:

Systems Editoriale
Milano

La Redazione,
per potenziare
la propria rete
di collaboratori,

CERCA ESPERTI

in possesso dei seguenti requisiti:

1- Possesso di almeno uno dei due computer che saranno oggetto di trattazioni su queste pagine (Ms-Dos compatibili / Amiga)
2- Conoscenza reale e approfondita di almeno due linguaggi di programmazione moderni (Turbo Pascal, Turbo C, Quick Basic, Quick C, Assembly 68XXX, Assembly 80X86).
3- Conoscenza reale e approfondita dell'uso di almeno un Word Processor per Ms - Dos e/o Amiga (è gradita la capacità di usare pacchetti D.T.P.).
4- Reale capacità di sviluppare programmi e articoli attenendosi strettamente alle indicazioni fornite dalla Redazione.

I candidati ideali sono residenti nell'Hinterland milanese; desiderano svolgere una collaborazione esterna, in completa autonomia; sono in grado di portare a termine articoli e programmi nei tempi stabiliti. Gli interessati possono contattare ***telefonicamente*** *la Redazione, allo scopo di fissare un appuntamento per l'esame dettagliato delle proposte redazionali e delle richieste degli aspiranti collaboratori.*

Per informazioni, tel: 02 / 57.60.63.10

di Michele Saladini

HARD COPY PER MPS 1230

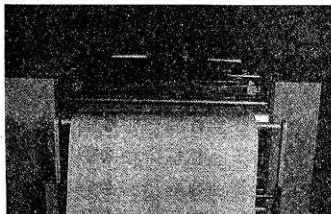
La stampante Commodore offre la possibilità di utilizzare numerosi comandi di tipo Esc; sfruttiamo tale possibilità con il C/64

La maggior parte delle stampanti attualmente in commercio offre la capacità di emulare altri modelli, specialmente la famosa **Epson FX**. Anche la **Mps - 1230** offre tale opportunità, rendendo così disponibile, per il **C/64**, una stampante di caratteristiche decisamente superiori al vecchio modello **803**.

Il programma che presentiamo sfrutta, appunto, l'**emulazione Epson FX** per ottenere rapidamente la riproduzione su carta di una pagina grafica. La differenza dalle normali routine di hard-copy consiste nell'opportunità di scegliere diversi tipi di densità, riuscendo così a stampare grafici con un rapporto degli assi di **1:1**, eliminando il fastidioso inconveniente che allungava orizzontalmente il disegno (trasformando, in altre parole, i cerchi in ellissi).

Il programma

Il programma Basic legge i dati che costituiscono la routine in l.m; nel caso non vi siano errori, viene direttamente creato, su disco, un file eseguibile lungo appena 2 blocchi, che verrà allocato a partire dalla locazione **49152**.

Il funzionamento del programma è molto semplice: si tratta di converire, byte per byte, i dati dello schermo nel formato richiesto dalla stampante. I valori che compongono la pagina grafica vengono rappresentati sullo schermo come linee **orizzontali**; al contrario, sappiamo che gli aghi della stampante sono disposti **verticalmente.** Il lavoro di "conversione" viene svolto dalla routine apparsa nel numero 62 nell'articolo "Da video a stampante" e che è parte integrante del programma di queste pagine.

La novità del listato consiste, inoltre, nel suggerire ai lettori che non dispongono di stampanti Commodore (ma di modelli di altre marche collegate al C/64 tramite interfaccia Centronics o di altro tipo) di tentare egualmente l'attivazione del listato. Infatti, dal momento che l'emulazione Epson è di tipo universale, il programma dovrebbe andar bene con qualsiasi tipo di printer in grado di emulare il popolare protocollo di attivazione della pagina grafica su carta.

Per attivare la modalità grafica bisogna inviare, alla stampante, i codici **27** (che corrisponde ad **Esc**), e **42** (che corrisponde all'**asterisco**) seguiti dal parametro della **densità** e da due byte (nel formato low / hight) indicanti il numero di dati che verranno interpretati come parte del grafico; in questo modo vengono stampate strisce alte **8** punti, e il singolo punto viene stampato se il corrispondente bit del byte è posto a 1 (il bit 7 attiva il primo ago, il numero 0 attiva l'ottavo; il nono ago non viene utilizzato).

All'inizio della procedura viene controllata l'eventuale presenza dei parametri e della loro correttezza; in caso di omissio-

ds	Modo	Punti / pollice	Punti / riga
0	Densita' normale	60	480
1	Doppia densita', meta' velocita'	120	960
2	Doppia densita', velocita' normale	120	960
3	Quadrupla densita'	240	1920
4	Video grafico I	80	640
5	Grafica plotter (x:y=1:1)	72	576
6	Video grafico II	90	720

Significato del parametro DS da indicare nel comando di Hard Copy

ne vengono assunti i valori di default. Successivamente, utilizzando le routine di **listen, second,** eccetera, viene aperto il file n. 1 ed inviata alla stampante la sequenza di Escape necessaria ad attivare la grafica in modalità **B.I.M.**, operazione che verrà effettuata all'inizio di ogni riga.

A questo punto viene chiamata la routine principale (quella apparsa su C.C.C. n. 62) incaricata di stampare gli 8 byte convertendoli nel formato richiesto: con l'ausilio di una tabella viene preso il primo bit di ogni byte del gruppo, ottenendo il valore da inviare al canale d'uscita; viene poi ripetuta la procedura con i bit successivi ed incrementati i puntatori al prossimo gruppo di 8 byte. Una volta completata la stampa viene ristabilita l'interlinea originale (precedentemente posta a **8/72"**, per ovvi motivi) e chiuso il file.

La sintassi

La sintassi che dovrete usare dopo aver lanciato il programma Basic è:

Sys 49152 [,Loc [,Ds]]

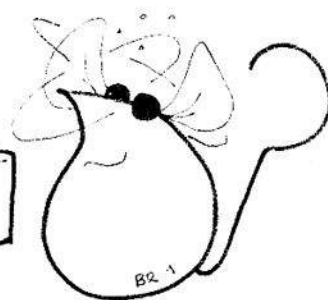
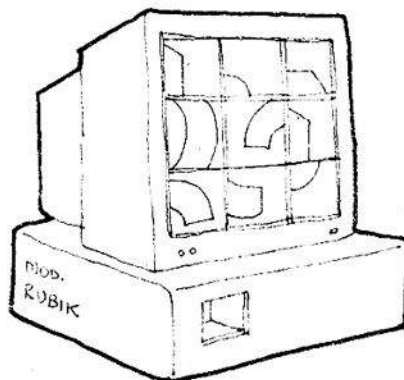
...in cui **Loc** indica la locazione di inizio della pagina grafica (possono essere stampate anche le schermate posizionate sotto le Rom, come quelle del Gw Basic e del Simons' Basic), e **DS** indica la densità di stampa (vedi tabella); i valori di default sono, rispettivamente, **8192** e **5**.

N.B: non bisogna premere il tasto **Restore** mentre la stampa è in esecuzione; le Rom, infatti, risultano disabilitate in tale frangente ed il computer si blocca.

```
100 rem *----- hard copy  v1.0 -----*
110 rem per mps 1230 in modo epson fx
120 rem by saladini michele    1991
130 rem *--------------------------*
140 :
150 ck=35438:s=49152
160 reada:ifa<>-1thenc=c+a:goto160
170 ifck<>cthenprint"errore nei dati":end
180 print"dati ok":input"nome file";a$
190 a$=a$+",p":open15,8,15:open1,8,1,a$
200 input#15,a,b$:ifathenprint"disk error:";a;b$:close1:close15:end
210 print#1,chr$(0);chr$(192);:restore
220 reada:ifa<>-1thenprint#1,chr$(a);:pokes,a:s=s+1:goto220
230 close1:input#15,a,b$:printa;b$:close15:print"file chiuso"
240 print"■uso: sys 49152,loc,tipo"
250 print"loc=locazione di inizio schermo grafico"
260 print"tipo= tipo di stampa"
270 end
5000 data032,121,000,201,044,208,036,032,253,174,032,138,173,032,247
5010 data183,165,020,133,251,165,021,133,252,032,121,000,201,044,208
5020 data020,032,241,183,224,007,048,015,162,014,108,000,003,169,000
5030 data133,251,169,032,133,252,162,005,134,254,024,165,251,105,064
5040 data141,098,192,165,252,105,031,141,104,192,032,128,192,032,191
5050 data192,169,000,133,253,032,217,192,032,140,192,024,165,251,105
5060 data008,133,251,240,015,165,251,201,064,208,011,165,252,201,063
5070 data208,005,076,003,193,230,252,230,253,165,253,201,040,208,219
5080 data169,013,032,168,255,076,076,192,169,001,162,008,157,059,003
5090 data010,202,208,249,096,169,000,170,168,224,008,240,042,192,008
5100 data240,024,032,243,192,177,251,032,252,192,061,060,003,240,008
5110 data165,002,024,121,060,003,133,002,200,208,228,160,000,165,002
5120 data032,168,255,169,000,133,002,232,208,210,096,169,004,032,177
5130 data255,169,097,032,147,255,169,027,032,168,255,169,065,032,168
5140 data255,169,008,032,168,255,096,169,027,032,168,255,169,042,032
5150 data168,255,165,254,032,168,255,169,064,032,168,255,169,001,032
5160 data168,255,096,120,169,052,133,001,173,000,160,096,072,169,055
5170 data133,001,104,096,169,013,032,168,255,169,027,032,168,255,169
5180 data050,032,168,255,032,174,255,096,-1
5190 end

ready.
```

```
   ;disassemblato commentato routine hard copy

        org $c000
start   jsr $0079               ;legge un carattere
        cmp #$2c                ;se non e' una virgola
        bne start1              ;vai a start1
        jsr $aefd               ;altrimenti la salta
        jsr $ad8a               ;e legge il
        jsr $b7f7               ;dato successivo
        lda $14                 ;e lo memorizza
        sta $fb                 ;in $14 - $15
        lda $15                 ;nel formato
        sta $fc                 ;low/hight
        jsr $0079               ;legge il prossimo carattere
        cmp #$2c                ;se non e' una virgola
        bne start2              ;vai a start2
        jsr $b7f1               ;atrimenti legge il dato
        cpx #$07                ;se e' < di 7
        bmi start2+2            ;lo memorizza
        ldx #$0e                ;altrimenti stampa
        jmp ($0300)             ;illegal quantity
start1  lda #$00                ;mette i valori
        sta $fb                 ;di default
        lda #$20                ;per l'inizio della
        sta $fc                 ;pagina grafica
start2  ldx #$05                ;valore di default
        stx $fe                 ;per il tipo di stampa
        clc                     ;azzera il carry
        lda $fb                 ;calcola l'indirizzo finale
        adc #$40                ;della pagina grafica
        sta l+1                 ;sommando 8000 ai
        lda $fc                 ;valori iniziali e
        adc #$1f                ;memorizza il risultato
        sta h+1                 ;all'interno del programma
        jsr tab                 ;crea una tabella
        jsr open                ;apre il file
prg     lda #$00                ;azzera il contatore
        sta $fd                 ;di fine riga
        jsr esc                 ;invia la sequenza di escape
prg1    jsr print               ;stampa 8 byte
        clc                     ;azzera il carry
        lda $fb                 ;incrementa il
        adc #$08                ;puntatore low
        sta $fb                 ;di 8 byte
        beq prg2                ;se non c'e' un riporto vai a prg2
        lda $fb                 ;altrimenti controlla se il
l       cmp #$40                ;byte basso corrisponde con il valore finale
        bne prg3                ;no: vai a prg3
        lda $fc                 ;si: controlla anche
h       cmp #$3f                ;il byte alto
        bne prg3                ;se diverso vai a prg3
        jmp close               ;se uguale chiude il file
prg2    inc $fc                 ;incrementa il byte alto
prg3    inc $fd                 ;incrementa il puntatore di fine linea
```

```
        lda $fd          ;se la linea corrente
        cmp #$28         ;non e' terminata
        bne prg1         ;vai a prg1
        lda #$0d         ;se lo e' manda un
        jsr ciout        ;return alla stampante
        jmp prg          ;vai a prg
tab     lda #$01                  ;crea in
        ldx #$08                  ;$033b - $0343
tab1    sta $033b,x               ;una tabella decrescente
        asl a                     ;necessaria per calcolare
        dex                       ;il byte da inviare
        bne tab1                  ;alla stmpante



        rts                       ;fine routine
stampa il carattere puntato da fb-fc        ;
print   lda #$00                  ;azzera gli indici
        tax                       ;A X Y per i
        tay                       ;cicli
print1  cpx #$08                  ;se ha stampato un byte
        beq print5                ;va alla fine
print2  cpy #$08                  ;controlla se ha finito la colonna
        beq print4                ;stampa il byte
        jsr ram                   ;altrimenti attiva la ram
        lda ($fb),y               ;legge il byte
        jsr rom                   ;riattiva la rom
        and $033c,x               ;estrae il bit interessato
        beq print3                ;se e' nullo va a print 3
        lda $02                   ;se e' attivo somma
        clc                       ;al contenuto di $02
        adc $033c,y               ;il valore corristpondente
        sta $02                   ;a quel bit
print3  iny                       ;incrementa y
        bne print2                ;salta all'indietro
print4  ldy #$00                  ;azzera y
        lda $02                   ;legge il byte da stampare
        jsr ciout                 ;e lo invia alla stampante
        lda #$00                  ;pulisce la loc. 2
        sta $02                   ;per il prossimo ciclo
        inx                       ;incrementa x
        bne print1                ;ricomincia
print5  rts                       ;fine routine
open    lda #$04                  ;prepara la periferica
        jsr listen                ;numero 4
        lda #$61                  ;apre il file
        jsr second                ;numero 1
        lda #$1b                  ;manda alla stampante
        jsr ciout                 ;il codice di esc
        lda #$41                  ;per avere un'interlinea
        jsr ciout                 ;di 8/72"
        lda #$08                  ;senza spazi vuoti
        jsr ciout                 ;tra le linee
        rts                       ;ritorna
```

```
esc     lda #$1b            ;invia alla stampante
        jsr ciout           ;
        lda #$2a            ;
        jsr ciout           ;la sequenza di esc per la grafica
        lda $fe             ;
        jsr ciout           ;in modo BIM di 320 dots
        lda #$40            ;
        jsr ciout           ;il valore per il tipo di
        lda #$01            ;
        jsr ciout           ;stampa e' contenuto nella loc $fe
        rts                 ;
ram     sei                 ;disabilita interrupts

        lda #$34            ;mette 52 nella loc 1
        sta $01             ;per disattivare le ROM
        lda $a000           ;legge un byte per attivare la RAM
        rts                 ;ritorna
rom     pha                 ;salva l'accumulatore
        lda #$37            ;mette 55 nella loc 1
        sta $01             ;per riattivare le ROM
        pla                 ;recupera A
        rts                 ;ritorna
close   lda #$0d            ;va a capo e
        jsr ciout           ;
        lda #$1b            ;
        jsr ciout           ;rimette l'interlinea originale
        lda #$32            ;
        jsr ciout           ;



        jsr unlsn           ;chiude il file
        rts                 ;ritorna

labels:
listen equ $ffb1
second equ $ff93
cioot  equ $ffa8
unlsn  equ $ffae
```

di Francesco Varone

VIAGGIO NELLE ICONE DI GEOS

Le icone di Geos 64: dove sono, come sono fatte, come modificarle

L'articolo che state leggendo non è l'ennesimo scritto per l'Amiga, come lascia immaginare il titolo. Infatti è dedicato ai tradizonali, affezionati (ostinati?) utenti del Commodore 64, in particolare del Geos.

Il **Geos**, "nuovo" Sistema Operativo del C/64, oltre ad offrire una maggiore velocità negli accessi al drive, ed una standardizzazione dei dati (cioè possibilità di fondere lavori di varie applicazioni in un unico documento), comunica con l'utente grazie ad icone, ovvero piccole immagini rappresentanti particolari funzioni raggiungibili con il joystick o con il mouse, in modo da rendere l'utilizzo dei programmi molto semplice ed intuitivo.

Alcuni tipi di icone, ed esattamente quelle associate ai file, sono leggibili e modificabili grazie a particolari tecniche di accesso al drive.

I file - icone

Come molti avranno già immaginato, i file che rappresentano le icone sono comunissimi **sprite** gestiti dal DeskTop sul taccuino della directory ed i dati ad esse relativi hanno il formato descritto sul manuale del C/64: ad ogni punto corrisponde un bit; uno sprite è lungo 24 bit e alto 21; per cui 63 byte (= 504 bit) definiscono un oggetto, ma su questo argomento non ci soffermiamo a lungo.

Per esaminare dove, sul floppy, vengono memorizzate le icone, sarà opportuno riferirsi al riquadro specifico.

Icone, come raggiungerle

In pratica, per osservare un blocco info, è necessario sapere dove esso si trovi sul dischetto e leggerne l'allocazione dalla directory. Vi sono vari metodi per leggere il direttorio (banale traduzione di directory):

1) impartire in modo diretto il comando **Load "$", 8** e quindi **List**;

2) aprire un file sequenziale con **Open Filenum, Device, Canale, "$"** (con canale diverso da **0**, altrimenti otterremmo lo stesso risultato del punto 1), e interpretare i vari byte caricati con **Get#** (un programma che utilizza questa tecnica è descritto sul manuale del drive 1541);

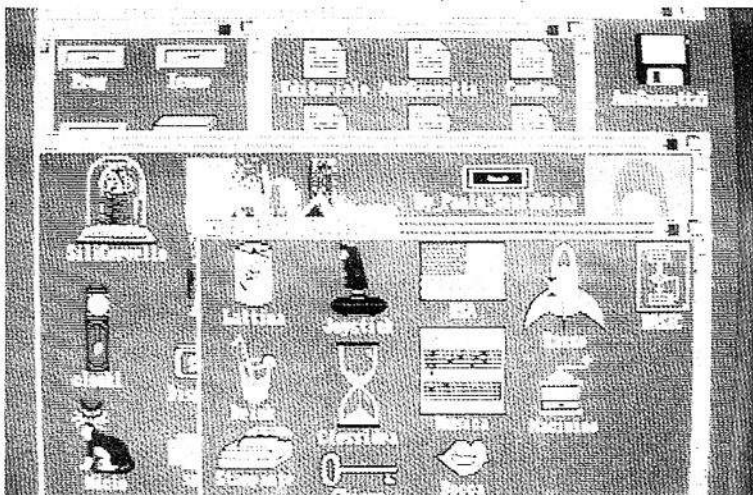
3) utilizzare l'accesso casuale (o diretto, che permette di leggere un determinato blocco del dischetto indipendentemente dalla sua posizione) con **Open Filenum, Device, Canale, "#"**, sapendo come ottenere i dati in modo opportuno.

Quale metodo utilizzare?

Scartiamo il primo, perchè fornisce solo i dati essenziali, cioè quelli che leggiamo normalmente prima di caricare un programma, e non consente elaborazioni.

Il secondo potrebbe, invece, essere utile. Tuttavia il sistema più comodo è senza dubbio il terzo, molto più immeditato e diretto in quanto, tra l'altro, alla fine della lettura della directory, troveremo già a nostra disposizione un file casuale che carica anche il blocco info.

Aperto, ad esempio, con **Open 5, 8, 5, "#"** un file casuale e con **Open 15, 8, 15** il canale di comando per gestirlo, bisogna innanzi tutto scoprire in quale blocco ed in quale campo della directory sia situato il nome del file da esaminare. Per questo scopo non c'è altra soluzione che confrontare, uno per uno, tutti i nomi dell'elenco della directory con quello che interessa (cfr. righe 100 - 200 del listato). Impartendo, attraverso il canale di comando...

PRINT #15, "U1:"; 5; 0; 18; s

...con **S** che indica il settore e **18** la traccia, si comunica al Dos di caricare il blocco della directory contenente il file in un buffer del drive (queste istruzioni devono essere inserite in un programma; utilizzate, comunque, il listato pubblicato come esempio, da confrontare con questi suggerimenti). Quindi con i comandi:

Print #15, "b-p:"; 5; 19 + 2 + 32 * C
Get#5, T$, S$
T = Asc (t$ + Chr$(0))
S = Asc (s$ + Chr$(0))

...(in cui la prima istruzione è il Buffer pointer; **C** indica il campo della directory; **19** è la locazione all'interno del campo) posizioniamo il puntatore di lettura / scrittura sul byte 19 e da lì leggiamo i dati consecutivi per venire a conoscenza della posizione del blocco info: **T** = traccia e **S** = settore. Caricandolo in un buffer del drive con...

Print#15, "U1:"; 5; 0; T; S

...avremo finalmente a disposizione i dati dell'icona che possiamo caricare, a partire dal byte 5, con:

Print#15, "b-p:"; 5; 5
For A = 0 To 63: Get#5, A$
Poke 832, Asc (a$ + Chr$(0)): Next

Il precedente gruppo di istruzioni inserisce i dati dell'icona nel **blocco 13 degli sprite** (vedi manuale, capitolo 6). In seguito si può visualizzare lo sprite e svolgere qualsiasi operazione possibile.

Approfittiamo

Sicuramente i più desiderosi di fama e gloria avranno notato, nella tabella 3, la presenza dell'autore del file. E' possibile, infatti, leggere o modificare anche questo.

Per fare ciò, dopo aver seguito tutti i passi prima descritti, posizionate il puntatore con:

Print #15, "B-P:"; 5; 77

Tabella 2
Campo In Formato Geos

0	Tipo file standard (PRG, SEQ etc.)
1- 2	T&S primo blocco dati
3-18	Nome file con spazi shiftati
19-20	T&S blocco info
21	Struttura file (0 = SEQUENTIAL, 1 = VLIR)
22	Tipo file GEOS (v. Tabella 4)
23-25	Data (Anno / mese / giorno)
26-27	Ora (Ora / minuti)
28-29	Lunghezza in blocchi

Per leggere i dati, ora, **non** si può usare il comando **Input#**, perchè la stringa termina con Chr$ (0) e non con Chr$ (13): dovete quindi utilizzare un loop contenente Get#, come questo:

999 ...
1000 N$ = ""
1010 Get A$:If A$=Chr$(0)Then 1030
1020 N$ = N$ + A$: Goto 1010
1030 ...

Per scrivere è più semplice; basta:

Print#5, Nomeautore$; Chr$ (0)

Notate, però, che il nome dell'autore compare nell'opzione info del menu file soltanto nei file di tipo Basic Program, Assembly Program, Desk Accessory e Application.

Inoltre non bisogna dimenticare che Geos gestisce i testi con un codice molto simile al codice Ascii, ma che non è il Petascii del C/64; ne consegue che prima di utilizzare una stringa bisogna convertirla (listato righe 600 / 640).

Ciò vale anche per il nome: poichè il programma lo chiede in formato Geos, bisogna immetterlo come tale; in realtà non bisogna fare altro che invertire le maiuscole in minuscole e viceversa op-

0	Tipo file (PRG, SEQ etc.)
1- 2	T&S primo blocco dati
3-18	Nome file con spazi shiftati
19-20	T&S primo side-sector (solo file REL)
21	Lunghezza record (solo file REL)
22-25	Non usati
26-27	T&S di sostituzione file con OPEN @
28-29	Lunghezza file in blocchi

Tabella 1
Singolo Campo Della Directory

0 - 1	T&S del link (0, 255: no blocco)	
2 - 4	3, 21, 191: dati riconosc.	
5 - 67		Dati sprite
68 - 76		Non usati
77 -		Classe file (*)
97 -		Autore (*)
160 -		Messaggio (*)

(*)ASCII terminante con CHR$(0)

Tabella 3
Blocco Info Geos

0	Non-GEOS File	6	Application (*)
1	Basic Program*	7	Application Data
2	Assembly Prg.*	8	Font File
3	Data File	9	Printer Driver
4	System File	10	Input Driver
5	Desk Accessory		

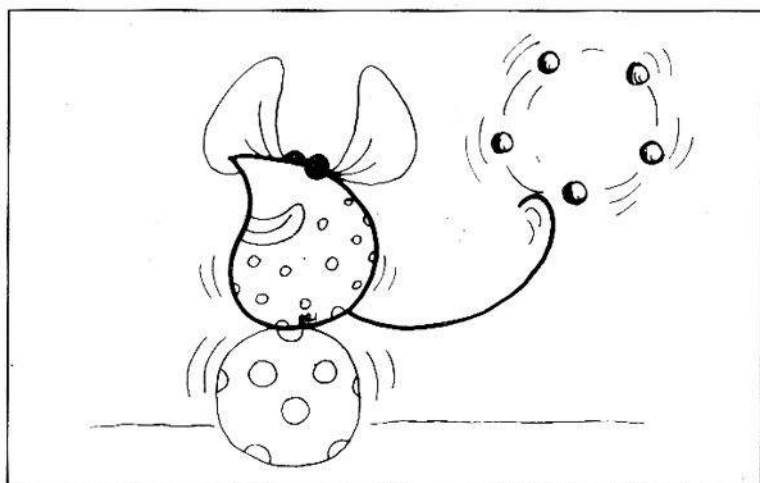
(*) inseribile autore

Tabella 4
Tipo File Geos

pure, se vi risulta difficile(?), leggete il nome giusto con il DeskTop del Geos!

Fantasia

Senza dilungarci troppo, ricorderemo che, per inserire l'icona in un file non Geos, bisogna aggiungere, nel campo file, le informazioni descritte nella tabella 2, e creare un blocco contenente i dati di tabella 3; evitate di convertire file RELativi in quanto i vostri GEOdati potrebbero creare conflitti con le informazioni necessarie alla gestione da parte del DOS.

Ad ogni modo, a meno che non siate dei campioni del C/64 (in questo caso crediamo che già sappiate tutto!), evitate di fare pasticci su dischi importanti. Usate piuttosto un programma specifico...

Struttura di un floppy

Il disk drive 1541, durante la formattazione, divide il floppy in **35** circonferenze concentriche, dette tracce, divise a loro volta in un numero variabile di blocchi o settori, in numero minore a mano a mano che si procede verso il centro del supporto stesso: **21 blocchi** nella traccia più esterna (la numero 1), e **17** in quella più interna (numero 35). I settori sono gruppi di dati di **256 byte** (in effetti sono in numero maggiore, se si considerano anche i SYNCronizzatori, gli IDentificatori, le checksum ecc., riservati al DOS e non facilmente raggiungibili dall'utente). I primi due byte di ogni blocco, che rappresentano il cosiddetto **link**, indicano, rispettivamente, il numero di traccia e di settore del successivo blocco.

La traccia centrale è, nell'uso standard, adibita alla memorizzazione della directory, elenco dei file con le varie caratteristiche tecniche (lunghezza, tipo, traccia e settore di partenza ecc.). Di questa traccia, il settore 0 contiene l'intestazione e la **BAM** (**B**lock **A**vailability **M**ap, mappa della disponibilità dei blocchi) del disco. I successivi, collegati dal link, sono divisi ciascuno in otto campi (o entry) di 32 byte, contenenti i dati indicati in tabella 1.

La Directory di Geos

Come si può notare dalla tabella 1, non tutti i byte di un campo sono sempre utilizzati.

Geos sfrutta, appunto, questo spazio per memorizzare altre informazioni, come indicato nella tabella 2. In particolare, bisogna notare il contenuto dei byte **19** e **20**. Infatti essi indicano la traccia e il settore (**T & S** per gli amici) in cui è memorizzato il blocco di informazioni.

Questo particolare settore contiene alcuni messaggi che potete leggere attivando un'icona e selezionando l'opzione **info** dal menu file.

Ed è proprio qui che l'icona è memorizzata.

Il programma pubblicato intercetta il blocco info di un file compatibile Geos e carica l'icona in uno sprite. Ora è possibile osservarla semplicemente in doppia altezza e doppia larghezza, oppure ricopiare i dati con:

```
For A = 0 To 63:
Print Peek (832 + A),: Next
```

oppure, per la stampante:

```
Open 1, 4: For A = 0 To 63:
Print#1, Peek (832 + A),: Next
```

...e magari inserirli in propri programmi o sostituirli nel demo della mongolfiera volante presente nel manuale.

```
10 rem       icon reader
20 rem  by varone francesco
30 rem (c) 1991 by compuclub
40 rem
50 print chr$(147);chr$(14);tab(14);"ICON  READER"
60 print:input"Nome file (Formato GEOS)";n$
70 n$=left$(n$,16):gosub 600
80 print"Inserisci il floppy e ";:gosub500
90 rem scansione directory
100 open 15,8,15,"i":open 5,8,5,"#"
110 t=18:s=1:print"Ricerca..."
120 print#15,"u1:";5;0;t;s:print#15,"b-p:5,0"
130 get#5,t$,s$:rem prossimo blocco
140 for c=0 to 7:m$="":print#15,"b-p:";5;2+c*32
150 get#5,a$,b$,b$:tp=asc(a$+chr$(0))and7:if tp=0 or tp=4 then 190
160 for a=1 to 16:get#5,a$:if a$=chr$(160) then a=16:goto 180
170 m$=m$+a$
180 next a:if m$=n$ then 210
190 next c:if t$<>"" then t=asc(t$):s=asc(s$+chr$(0)):goto 120
200 print"File non trovato.":print:goto 540
210 rem ricerca blocco info
220 print#15,"b-p:"5;21+c*32:get#5,t$,s$
230 t=asc(t$+chr$(0)):s=asc(s$+chr$(0))
240 if t=0 or t>35 then print"File non formato GEOS.":goto 540
250 print#15,"u1:";5;0;t;s:print#15,"b-p:5,5"
260 rem caricamento e
270 rem visualizzazione sprite
280 for a=0 to 63:get#5,a$
290 poke832+a,asc(a$+chr$(0)):next
300 v=53248:poke 2040,13:poke v+16,1
310 poke v,40:poke v+1,60:poke v+21,1
320 poke v+23,1:poke v+29,1:poke v+39,1
330 gosub 600:print chr$(147);"Icona del file ";n$
340 gosub 500:poke v+21,0
350 run
360 :
500 rem premi un tasto
510 print"Premi un tasto.":print
520 get a$:if a$="" then 520
530 return
540 rem restart
550 gosub 500:run
600 rem conversione stringa
610 m$="":for a=1 to len(n$):a$=mid$(n$,a,1)
620 if a$>"@" and a$<="z" then a$=chr$(asc(a$)+32)
630 if a$>="A" and a$<="Z" then a$=chr$(asc(a$)-128)
640 m$=m$+a$:next:n$=m$:return
650 end
```

Borland TURBO PASCAL 5.5

Tra breve verrà commercializzata la versione 6 del potente compilatore Borland; diamo uno sguardo al linguaggio oggi più noto in ambiente Ms Dos

Nel novembre '89 veniva presentata, sulla rivista **Personal Computer**, la "nuova" versione **5.5** del compilatore Turbo Pascal, che presentava notevoli migliorie rispetto alla precedente versione **3.0**.

Dallo scorso dicembre la **Borland** ha iniziato la distribuzione della nuovissima versione **6.0** che rappresenta qualcosa di realmente stupefacente nel mondo dei linguaggi, sia grazie alle innovazioni che offre, sia per l'introduzione di adeguati accorgimenti in grado di sfruttare adeguatamente l'hardware delle nuove macchine.

Inutile dire che, almeno per il momento, è disponibile solo la versione inglese; per la traduzione dei manuali in italiano stanno comunque provvedendo in tutta fretta.

A che può servire, dunque, parlare oggi della versione di un linguaggio che rischia di diventare obsoleta in poco tempo? Semplice: anzitutto non molti, tra i nostri lettori, hanno finora avuto la possibilità di esaminare da vicino il contenuto dei manuali originali (**in italiano**, ovviamente); inoltre è sempre bene informarsi sulle potenzialità della versione commercializzata fino a questo momento per effettuare (quando parleremo diffusamente della versione 6) un confronto "diretto" sulle potenzialità vecchie e nuove del famoso compilatore.

Le caratteristiche

Descrivere un linguaggio di programmazione è un'impresa molto più ardua di quella affrontabile nella descrizione (tanto per fare un esempio) di un word processor. Chi conosce le potenzialità di un linguaggio non ha interesse a leggerne l'articolo descrittivo; viceversa, chi è abituato al solo **Gw Basic**, non capirebbe nulla di quanto viene descritto.

Ci limiteremo, quindi, a delineare le parti più salienti cercando di offrire una panoramica sufficientemente comprensibile, almeno a chi è dotato di buona volontà. Anzitutto, c'è da dire che un compilatore consente di realizzare un programma (**file oggetto**) in grado di essere lanciato direttamente da **Dos**, senza che sia necessaria la presenza, insomma, del compilatore stesso (con un programma in Gw Basic, come è noto, è invece necessaria la presenza dell'interprete).

Inoltre un compilatore, cosa importante, effettua una conversione da linguaggio ad alto livello in un linguaggio molto vicino al linguaggio macchina; di conseguenza la velocità di esecuzione aumenta considerevolmente, soprattutto se paragonata alla velocità di un linguaggio interprete.

Turbo Pascal consente, come già detto, il richiamo delle varie funzioni (**ambiente integrato**) grazie a finestre richiamabili con l'aiuto di un tasto. In qualsiasi momento potete invocare (quasi) tutte le funzioni disponibili. E' impossibile, infatti, cercare di far partire un programma che contiene errori; nè potete linkare file inesistenti o altre amenità di analogo, infimo livello...

La caratteristica più comoda per i programmatori è sicuramente costituita dall'**help in linea**, che con il T. Pascal, raggiunge i massimi livelli di versatilità. Vediamo come e perchè.

Ipertesto

Se state scrivendo un programma e non ricordate, ad esempio, come e quando potete usare un certo comando, è sufficiente posizionarvi **sulla** parola che rappresenta il comando e premere il tasto di aiuto: immediatamente viene aperta una finestra in cui appare una descrizione abbastanza approfondita delle potenzialità offerte dal comando stesso. Se non vi basta, è possibile far apparire un'altra finestra, in cui appaiono esempi pratici di micro-programmi che illustrano meglio l'uso del comando; non solo: con le funzioni taglia ed incolla potete addirittura "estrarre" uno spezzone del programma di esempio e "trascinarlo"

L'ambiente integrato

La caratteristica che accomuna i linguaggi delle nuove generazioni è certamente rappresentata dal cosiddetto **ambiente integrato**, temine che indica la possibilità di lavorare su un "oggetto" senza essere costretti ad usare una pluralità di programmi indipendenti gli uni dagli altri.

In parole un po' più semplici, ciò significa che, nel caso specifico di un listato da compilare, il programmatore deve, nell'ordine, effettuare varie operazioni, utilizzando, per ciascuna di esse, uno specifico programma (detto Tool di sviluppo):

- digitare il listato (editor)
- compilarlo (compiler)
- effettuare le inevitabili correzioni (debugger)
- apportare le modifiche (nuovamente editor)
- collegarlo con altri programmi (linker)
- individuare con facilità le cause degli errori (help)

Un ambiente integrato consente, **senza** "uscire" dall'applicazione, di effettuare tutte le operazioni citate, con un risparmio di tempo davvero notevole. Si potrebbe obiettare che, grazie alla velocità dei moderni computer, non dovrebbe costituire una difficoltà utilizzare un Editor, "abbandonarlo" temporaneamente, lanciare un Compiler e caricare nuovamente un editor per apportare le dovute modifiche.

Chi, però, sa che cosa significa lavorare in questo modo, non può trattenere un sorriso di compassione.

Come intuitivo, quindi, il Turbo Pascal rappresenta una meravigliosa sintesi di numerose opzioni, tutte richiamabili grazie ad altrettanti menu a tendina e, soprattutto, completamente automatiche e "trasparenti" per il programmatore che, in questo modo, può dedicare tutti i suoi sforzi alla stesura del programma che ha in mente.

all'interno del listato che state scrivendo, in modo da effettuare con comodo gli adattamenti del caso.

La struttura in cui è "ingabbiato", nel modo descritto, un qualunque file di testo, viene definita **ipertesto**. Per meglio chiarire le idee, immaginate di trovarvi di fronte ad un volume su cui è stampato un testo qualsiasi, ad esempio un canto della Divina Commedia: in calce, come è risaputo, sono riportate delle note in corrispondenza di specifici versetti (opportunamente numerati) oppure in corrispondenza di numeri affiancati, in piccolo, alle parole più ostiche. Ebbene, un qualsiasi testo scolastico della Divina Commedia rappresenta l'embrione di un ipertesto: è possibile, cioè, individuare note e spiegazioni corrispondenti a frasi o parole riportate nella pagina che si sta consultando.

Immaginate, ora, che tutte le pagine della Divina Commedia diventino **trasparenti** ad un vostro comando e che possa comparire, sovrapposta alla pagina che state consultando, una nota riportata dieci o trenta pagine più avanti (o più indietro) che può avere attinenza con quanto state leggendo in quel momento. Ad esempio: in quale altro canto Dante ha parlato di un certo Papa?, quali altri versetti contengono riferimenti alla città di Firenze? E così via. In pratica, un ipertesto è un misto di **vocabolario** e **database**, con possibilità di tagliare ed incollare eventuali parti di testo.

I manuali

Come spesso suggeriamo ai nostri lettori, soprattutto per ciò che riguarda i linguaggi, il possesso dei manuali originali è assolutamente **indispensabile** per affrontare **seriamente** un qualsiasi linguaggio. Non ci troviamo più, insomma, di fronte alla cartuccia del **C/64** che aggiunge al Basic una dozzina di nuovi comandi; le fotocopie delle sei o sette pagine dell'opuscolo (chiamarlo manuale sarebbe eccessivo) allegato alla cartuccia del C/64 potevano accontentare i pirati della precedente generazione tecnologica. Illudersi, oggi, di impadronirsi di un linguaggio con i "si dice" o, peggio, con fotocopie rubacchiate un po' dovunque, è come darsi la zappa sui piedi.

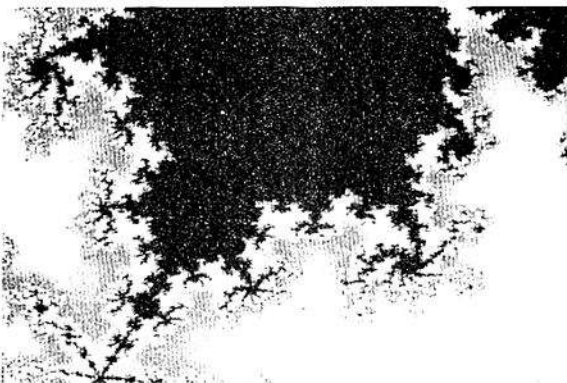
Gli studenti delle scuole superiori (che posseggono un libro di testo specifico per il T. Pascal) sanno benissimo ciò che diciamo dal momento che, in varie parti del volume, prima o poi ci si imbatte in messaggi del tipo... *"per approfondire l'argomento si consiglia di consultare i manuali acclusi alle confezioni originali Borland"*. Del resto non potrebbe esser diversamente: nessuna casa editrice oserebbe riportare brani pubblicati sui manuali originali, per ovvi motivi di Copyright.

L'acquisto della confezione originale porta, inoltre, numerosi altri vantaggi: ci limitiamo a citare la possibilità di ottenere consulenze telefoniche gratuite e quella di acquistare la versione più recente del linguaggio pagando una minima differenza di prezzo.

I manuali di cui ci occupiamo in queste pagine sono, ovviamente, quelli contenuti nella confezione originale Borland, **versione italiana** del T. Pascal 5.5.

Guida all'uso

Il volume, di oltre **360 pagine**, consta di otto capitoli, un'introduzione, sei appendici, un utile glossario, ed un indispensabile indice analitico (oltre a 18 figure e 15 tabelle).

La breve introduzione descrive sommariamente che cosa è il T. Pascal V.5.5,

quali sono le principali caratteristiche (e differenze rispetto alle precedenti versioni) ed indica le convenzioni tipografiche usate nei volumi.

Il **primo** capitolo consiglia quale compilatore usare, come riportare il contenuto dei floppy sul disco rigido e come usare il compilatore sui computer privi di hard disk (il disco rigido, dimenticavamo di dire, **non** è indispensabile per usare T. Pascal).

Il **secondo** capitolo descrive il modo di operare dell'ambiente integrato, che cosa sono gli help in linea (e come attivarli) e come usare correttamente le numerose finestre ed i menu disponibili. Questo capitolo sarà particolarmente apprezzato dai super-principianti dal momento che riporta i primi, brevissimi programmi da digitare (e lanciare) allo scopo di imparare l'uso delle numerose potenzialità di T. Pascal (digitazione, salvataggio, compilazione, eccetera). Viene anche descritto l'utilizzo della finestra Watch e delle funzioni Make e Build.

Dal **terzo** capitolo inizia lo studio vero e proprio del linguaggio: si passano in rassegna i sette elementi base della programmazione (input, dati, algoritmo, output, esecuzione condizionale, cicli, subroutine) e si descrivono minuziosamente tutti gli elementi tipici del compilatore: dati interi, reali, stringa, carattere, identificatori, operatori, istruzione if, procedure, funzioni e così via.

Il **quarto** capitolo è dedicato ai moduli (potenti strumenti della programmazione strutturata) ed è pieno di casi pratici legati a semplici programmi di esempio.

Nel capitolo **quinto** viene esplicitamente affrontato il problema della programmazione strutturata; viene descritto il metodo di creare file di tipo Make e si descrivono in modo approfondito le opzioni Make e Build, già accennate in precedenza. Le direttive **Define** e **Undef** e la descrizione dei simboli predefiniti (tra cui Ver50, If... Else... Endif, ed altri) vengono affrontati in questa sede.

Il **sesto** capitolo è dedicato alle delicate fasi della compilazione (con descrizione di errori di ogni tipo ed uso corretto del Debugger), dei punti di arresto e dell'individuazione di procedure, funzioni ed errori.

L'ambiente integrato è descritto nel **settimo** capitolo ed affronta tutti i problemi che possono presentarsi in fase di utilizzo dei menu che appaiono nelle finestre principali (File, Edit, Run, Compile, Options, Debug, Break/Watch).

Nell'**ottavo** (ed ultimo) capitolo si passano in rassegna le opzioni selezionabili in fase di compilazione (Switch directive, Conditional defines, Build all, Quiet, Include directory, Map file e tantissime altre).

Le sei **appendici** affrontano, rispettivamente, le differenze tra la versione 5.5 e le precedenti; l'uso dell'editor; il corretto utilizzo delle utility disponibili (Tpumover, Make, Macro, Touch, Grep ed altre); personalizzazione del T. Pascal (uso evoluto di Tinst ed eventuale modifica delle opzioni Compile, Options, Debug, Editor, Display, Colors, Windows, Quit/save); introduzione al Dos(!) e glossario.

Guida di riferimento

Se le pagine del volume precedente vi sembravano troppe, tenetevi forte: stavolta si tratta di oltre **500 pagine**, piene zeppe di notizie ed informazioni per l'**uso evoluto** e professionale del Turbo Pascal (capita l'importanza dei manuali originali?).

Oltre all'introduzione, il volume ospita ben 16 capitoli, 4 appendici ed un indice alfabetico dei contenuti.

Inutile dire che è proprio questo il libro(ne) che sarà fonte di sollazzo per i più esigenti programmatori. Ogni potenzialità del compilatore viene esaminata fino in fondo ed illustrata attraverso esempi chiarissimi ed alla portata di chiunque voglia affrontare lo studio del linguaggio in modo serio.

In questo paragrafo non ci dilungheremo nella descrizione del contenuto del manuale; ci basterà dire che gli argomenti evidenziati nel precedente volume vengono affrontati, approfonditi ed eventualmente correlati ad altri argomenti, sempre per mezzo di esempi e delucidazioni.

Un intero capitolo, ad esempio, viene dedicato alle costanti; un altro, poi, è dedicato all'uso del coprocessore matematico o alla sua emulazione; quello più importante, comunque, ci è sembrato il 16-mo che, in ordine alfabetico, passa in rassegna tutte le procedure e le funzioni disponibili in ambiente Turbo Pascal 5.5; vi basterà sapere che, da solo, occupa "appena" 195 pagine...

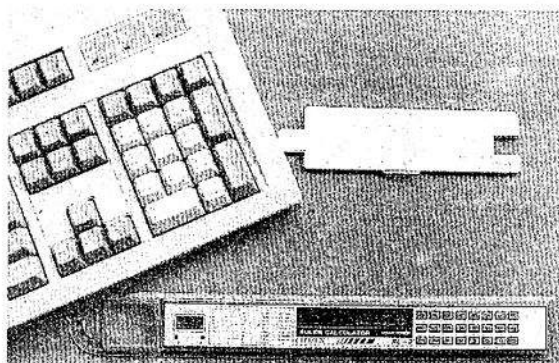
Approfondimenti sugli errori (e sui messaggi prodotti), confronti tra T. Pascal Borland e **Pascal Ansi** e direttive sul compilatore, completano le trattazioni affrontate dal voluminoso manuale.

Gli "oggetti"

Il terzo volume è di quasi 130 pagine ed è dedicato a chi vuole affrontare lo studio del T. Pascal 5.5 fino alle estreme possibilità. Si tratta, infatti, della **programmazione orientata agli oggetti**, peculiarità di notevole interesse offerta dai linguaggi di altissimo livello.

Inutile dilungarsi sul contenuto del volume: è roba per super esperti, già conoscitori del compilatore.

Basterà dire che, con la programmazione orientata agli oggetti, è possibile scrivere moduli di programmi che sono in grado di richiamare funzioni di altri moduli, precedentemente memorizzati, di cui si conoscono poche ma essenziali caratteristiche. Questa approssimativa definizione può far sorridere chi conosce adeguatamente **OOP** (**O**bject **O**riented **Pro**gramming); ma, credeteci, in poche parole non si può dire nulla di più.

Per maggiori informazioni:
Borland Italia
Via Cavalcanti, 5
20127 Milano
Tel. 02/26.10.10.2

di Carlo d'Ippolito

DISTRICARSI TRA I COMANDI

Alcuni comandi Ms-Dos sembrano inutili, o di importanza decisamente limitata. Vediamo di "ampliare" le loro applicazioni

Quando si scrive un articolo sui comandi di un sistema operativo, ci si rende conto che non è possibile affrontare un nuovo argomento senza aver prima chiarito determinati concetti.

Questi, a loro volta, richiedono precisazioni che tirano in ballo altre considerazioni, e così via, (quasi) all'infinito.

L'alternativa ad una inevitabile opera enciclopedica, che partirebbe dal bit per arrivare al Dos, è rappresentata dall'arido elenco dei numerosi comandi che, però, è già contenuto nel manuale di istruzioni del computer.

C'è, tuttavia, un altro modo di affrontare i vari argomenti ed è rappresentato, paradossalmente, dall'intrecciarsi disordinato di varie applicazioni che, prese singolarmente, sembrerebbero di ben scarsa utilità.

Nel presente articolo, pertanto, descriveremo le opportunità offerte da alcuni comandi, che, combinati insieme, offrono spunti di riflessione non descritti, sui comuni manuali, con sufficiente ampiezza.

Nel corso di queste note ci limiteremo a ribadire alcuni concetti riguardanti i comandi trattati; al lettore, come di consueto, è affidato il compito di esaminare il manuale **Ms - Dos**, che certamente possiede, per individuare le varie forme sintattiche e le altre notizie che, in questa sede, non riportiamo per evitare di ripetere concetti già noti(!).

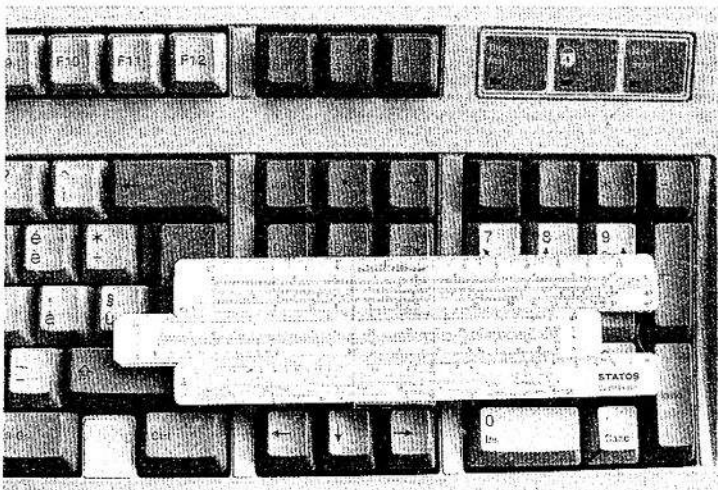
I file Batch

Anche il principiante dovrebbe sapere che un **file batch** non è altro che un file Ascii e, come tale, esaminabile con il comando **Type** o con un qualsiasi word processor o editor di linea. Il suo contenuto rappresenta un insieme di comandi Ms-Dos che verranno eseguiti, in successione, al momento della sua attivazio-

Scrivi anche tu!

Sei in grado di scrivere semplici, ma altrettanto efficaci, files di tipo Batch? Ritieni che, sperimentando in base ai nostri suggerimenti, tu sia pervenuto ad un risultato degno di nota?

Bene, invia il file batch su dischetto, non dimenticando di memorizzarvi anche una breve descrizione del funzionamento del file stesso.

Agli autori dei listati giudicati degni di pubblicazione verranno riservati abbonamenti alle testate della Systems Editoriale.

ne. E' bene sottolineare che un file batch DEVE avere il suffisso **.BAT**, altrimenti, nonostante il suo contenuto sia sintatticamente corretto, non può essere "lanciato".

Se, infatti, digitiamo un comando completo di suffisso, il computer cerca subito di rintracciarlo (sul drive attivo, cioè su quello indicato a sinistra del cursore lampeggiante) e, se lo trova, lo *lancia* immediatamente.

Sappiamo, del resto, che è possibile far eseguire un comando limitandoci a digitarne il solo nome, privo di suffisso; è opportuno, tuttavia, ricordare alcune particolarità del Dos.

Non appena, infatti, si digita un comando (privo di suffisso) alla tastiera, il computer controlla subito se questo è un comando "interno" (come **Date, Time, Type**, eccetera). In caso negativo cerca di rintracciare, sul drive attualmente attivo, il comando dotato di suffisso **.COM**; se non lo trova, cerca allora il file con suffisso .EXE; l'ultimo tentativo è riservato alla ricerca di un file batch, purchè questo sia stato memorizzato con il suffisso **.BAT**.

Per convincersene è sufficiente eseguire, alla lettera, i vari passi che descriveremo; ne approfitteremo, anzi, per evidenziare ciò che non tutti sanno.

Supponiamo che il file idoneo per la tastiera del vostro computer sia **KEYBIT.COM**, ma anche in caso diverso sarà possibile seguire il ragionamento.

Sapete di certo, infatti, che, tentando di chiamare un altro file **Keyb** (come, ad esempio, KEYBFR.COM), pigiando i vari tasti otterreste caratteri diversi da ciò che vi aspettate.

Alcune versioni del Dos dispongono del comando **Keyb** dotato di una sintassi un po' particolare: consultate, gente, consultate e modificate, se necessario, quanto stiamo per dire, magari usando un altro comando del Dos....

Formattate, ora, un dischetto (attenzione: il suo contenuto andrà irrimediabilmente perso) e, dal vostro disco sistema (inserito nel drive **B:**), copiate un qualsiasi file di tastiera (Keyb ....) purchè **non** sia Keybit.com; supponendo che il file scelto per gli esperimenti sia **Keybfr.com** (tastiera francese), effettuate quattro operazioni di copia utilizzando altrettanti nomi, che qui di seguito indichiamo ...

A> Copy b: Keybfr.com Socrate.Com
A> Copy b: Keybfr.com Calliope.Exe
A> Copy b: Keybfr.com Carneade.Sys
A> Copy b: Keybfr.com Cratete

...in modo che, insomma, siano tutti diversi l'uno dall'altro. Al termine, copiate anche il "vostro" file (Keybit.com) con lo stesso nome (lasciando in pace, una volta tanto, i filosofi greci):

A> Copy b: Keybit.com

A questo punto dovrebbe esser chiaro che l'*unico* file idoneo per la vostra tastiera è Keybit.com. Gli altri quattro serviranno solo per capire, grazie alla imprecisa digitazione che genereranno, se sono stati caricati, oppure no.

Provate, finalmente, a digitare **Keybit.com** e, subito dopo, accertatevi che, in effetti, ciò che compare sul video è esattamente ciò che digitate. Ora lanciate **Socrate.com** due volte, sia completo di suffisso (Socrate.com, appunto) sia senza (Socrate + tasto return). In entrambi i casi, digitando a casaccio sulla tastiera, vi accorgerete che il file è stato lanciato "correttamente" (cioè... non c'è corrispondenza tra video e tastiera).

Attivate nuovamente Keybit sia per eliminare gli inconvenienti di digitazione, sia per ripristinare le condizioni iniziali.

Digitando, ora, **Calliope.exe** (anche qui, prima *con* e poi *senza* suffisso) vi accorgete che il file viene individuato e lanciato.

Se, però, tentate di lanciare **Carneade.sys** (con o senza suffisso) oppure **Cratete** (che avevate memorizzato privo di suffisso) avrete la sgradita sorpresa di non vederli attivare: compare, addirittura, il tipico messaggio visualizzato nel caso in cui il file risulti assente ("Comando o nome archivio non valido" e simili, a seconda del vostro Dos).

Dunque non è sufficiente che un file rappresenti un programma correttamente impostato; è necessario che il suo nome abbia un suffisso "compatibile" con il tipo di file stesso.

Generalizzando, ciò potrebbe spiegare perchè alcuni file, pur essendo presenti su alcuni vostri dischetti, non vengono attivati correttamente; è probabile che possediate una copia di file privo di suffisso, oppure incompatibile con quello.

Senza fare altri esempi (anzi, vi invitiamo a sperimentare per vostro conto) possiamo affermare che non è possibile attivare un file batch se il suo nome è privo del suffisso specifico (.BAT).

Provate, poi, ad impostare file aventi, come nome, quello di alcuni comandi interni (**Date, Time, Dir**...). Pur se questi, dopo la copia e/o la ridenominazione (con o senza suffissi), saranno fisicamente presenti sul dischetto, non sarà mai possibile attivarli. Incongruenze del Dos, che accetta nomi di file che non potranno mai essere eseguiti!

A chi ritiene di essere un ricercatore, tuttavia, comunichiamo che c'è un modo per attivare egualmente uno di questi files dal nome "illegale". Al primo che ci comunicherà la corretta procedura da seguire (si tratta di un comando...) verrà inviato un pacco dono di prodotti Systems.

Percorsi

Un'altra opportunità, che può rivelarsi utile, è sicuramente la procedura che, denominata **Pathname**, consente di ricercare un certo file in "zone" diverse delle memorie di massa a disposizione.

Quando, in altre parole, abbiamo la certezza che è presente un certo comando su uno dei dischi inseriti nei drive, ma non ricordiamo su quale di essi, il computer provvederà automaticamente a passare in rassegna le varie directory (e subdirectory) indicate nel comando.

Supponiamo, per chiarire, di possedere due drive (**A:** e **B:**) in cui sono inseriti altrettanti dischetti. Con il comando...

A> Path b:;a:

...comunicheremo l'ordine di esaminare dapprima la directory del drive B e, in caso di insuccesso, quella del drive A.

E' bene sottolineare ciò che non tutti i manuali riportano: in qualsiasi caso verrà dapprima esaminata la directory del drive **corrente**.

Se, quindi, vi trovate nel drive **C**, il file desiderato verrà cercato PRIMA in C, POI in B e PER ULTIMO, in A.

Un'altra cosa che è bene chiarire è che i comandi interni **non** vengono assoggettati al percorso; **Dir**, tanto per fare un esempio, visualizza solo la directory del drive corrente; analogamente ...

A> Type Euripide.Txt

...funzionerà solo se il file **Euripide.Txt** sarà presente sul drive A; la ricerca non verrà, quindi, effettuata anche sugli altri

drive. Questo fatto può essere piuttosto seccante, e comunque limita la versatilità del comando stesso.

Un modo piuttosto comodo per "estendere" le potenzialità del comando Type è quello di ricorrere ad un file batch da attivare "sotto" il percorso path (riferito a tutti i drive disponibili) che supporremo attivato in precedenza.

Eseguite...

A> Path a:;b:

...se, ovviamente, disponete dei due drive A e B. In questo modo qualsiasi comando esterno verrà esaminato sia su A sia su B prima di un eventuale insuccesso di ricerca.

C'è da ricordare che, se "siete" nel drive B, il file verrà ricercato dapprima nel drive corrente (cioè B:) e poi nel path; in altre parole, la ricerca del comando nel drive B può venire effettuata due volte; piccole inefficienze del Dos!

Ora, con un qualsiasi word processor (oppure con **Edlin** o, ancora, con **Copy con:**, come qui suggerito), memorizzate il seguente file batch, di nome **Scrivi.bat**:

```
A> copy con: scrivi.bat
echo off
b:
if exist %1 goto :vedo
a:
if not exist %1 goto :inesistente
:vedo
type %1
goto :fine
:inesistente
echo il file cercato non esiste
:fine
Ctrl Z
```

Vedremo, ora, di esaminarlo e commentarlo riga per riga. Chi non sa nemmeno che cosa sia un file-batch, tuttavia, è opportuno che legga, in proposito, il manuale del computer che possiede.

"Dentro" un file batch

Oltre ai vari comandi del Dos, un file batch può ospitare altri comandi, tra cui, fondamentali, quelli relativi ai salti condizionati.

Echo off consente di evitare inutili visualizzazioni, che potrebbero disturbare l'output su video.

Il secondo comando incontrato (un semplice **b:**) impone al computer di attivare il secondo drive.

La terza linea è piuttosto interessante: indica al computer di saltare alla subroutine di nome **:vedo** nel caso in cui esista un certo "qualcosa" denominato **%1**.

Con tale (forse strana) forma sintattica si indica, in Ms-Dos, il gruppo di caratteri battuti subito dopo il nome del file batch.

Digitando, ad esempio ...

A> Scrivi.bat Teocrito.txt

...il computer esaminerà, nella directory corrente, l'esistenza del file **Scrivi.bat**. Se non dovesse esistere, verrà attivato il drive b: (è in azione il path, ricordate?) e, stavolta, verrà lanciato.

Insomma, dopo aver eseguito i primi due comandi (**echo off, b:**), verrà cercato (IF EXIST) il file "Teocrito.txt" (%1) e, se trovato, il computer salterà (GOTO) alla subrotine specificata (:VEDO), presente in fondo allo stesso file batch.

Nel caso in cui l'evento non si verificasse, il computer attiverà l'altro drive (A:) e controllerà, stavolta, l'inesistenza (IF NOT EXIST) dello stesso file (%1) in tale drive. In questo caso (se, cioè, il file NON esiste) il salto (GOTO :INESISTENTE) visualizzerà l'opportuno messaggio (penultima riga del file batch).

In caso affermativo, invece, il ben noto comando (Type), "applicato" al nome %1, renderà visibile il contenuto di Teocrito.txt.

Con il semplice trucchetto illustrato è quindi possibile estendere la procedura Path anche agli **argomenti** dei comandi.

La manipolazione di Type, come intuitivo, è solo una delle opportunità che il lettore, ne siamo sicuri, sarà in grado di rendere disponibili per il suo computer.

❑❒

La redirezione

Altro caso di notevole interesse è la cosiddetta **redirezione**, vale a dire la possibilità di riversare su un file (da gestire, di solito, in formato Ascii) l'output di un certo comando.

L'esempio più banale, che normalmente viene illustrato sui manuali, è quello che scrive la directory in un file, da esaminare poi con Type.

Esempio:

A> Dir > Filippo

Dopo tale operazione, sarà disponibile, sul floppy A, il file Ascii contenente tutti i nomi della directory corrente dello stesso dischetto.

L'utilità che, poi, viene sempre suggerita, è circoscritta all'ordinamento alfabetico della directory stessa, crescente o decrescente, utilizzando il comando **Sort**; tutto qui.

Vedremo, invece, un interessante utilizzo della redirezione; come esempio abbiamo scelto un programma, di nome **Whereis**, che forse non tutti possegono.

Un programma analogo, dotato di sintassi lievemente diversa, è l'altrettanto noto **Find**.

Ricordiamo, ad ogni buon conto, che i file-utility di cui parliamo sono di pubblico dominio e risulta semplice procurarseli.

Digitando ...

A> Whereis.com Demetrio.com

...ad esempio, l'utility di cui parliamo visualizza l'eventuale percorso di subdirectory necessario per "raggiungere" il file indicato (Demetrio.com, nel nostro caso). Una risposta tipica può essere ...

A> \Filosofi\Greci\Aristote\Demetrio.com

...nel caso in cui il file si trova contenuto nella subdirectory di nome **Aristote** che,

a sua volta è all'interno di **Greci**, nidificata, quest'ultima, in **Filosofi**.

Sfruttando il file Whereis, la tecnica della redirezione, la sintassi dei file batch e la modifica del percorso (Path), è possibile creare, ad esempio, un file del genere:

A> Copy con Dove.bat
echo off
whereis.com %1> jolly.bat
jolly.bat
Ctrl Z

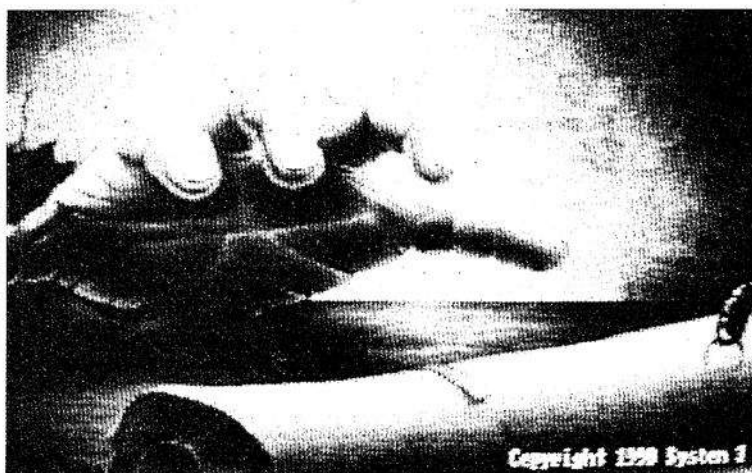
Evitando spiegazioni fin troppo ovvie (e ricordando che è ancora attivo **Path a:;b:**), ci limiteremo a sottolineare che, digitando ...

A> Dove.bat Demetrio.com

...verrà dapprima cercato il file Whereis.com sia nel drive A sia nel B (grazie al Path; in caso di insuccesso vedremo il solito messaggio).

L'utility, quindi, riverserà sul file di comodo "**Jolly.bat**" la stringa che contiene il percorso di subdirectory cercato.

Subito dopo, infine, verrà attivato proprio Jolly.bat che, guarda caso, guiderà il computer all'individuazione, e successiva attivazione, del comando Demetrio.com.

Concluderemo sottolineando il fatto che non vi deve essere uno spazio prima del carattere di maggiore (>) presente dopo %1. Ma questa è una peculiarità di Whereis (a proposito: questo comando significa "Dov'è ...?").

Conclusioni

Di certo il lettore sarà in grado di stendere file batch più efficienti e versatili di quelli, volutamente semplici, illustrati in queste pagine.

Non dobbiamo dimenticare, a nostra... discolpa, che la banalizzazione degli argomenti proposti non è casuale, ma voluta.

Aumenta sempre più, infatti, la schiera di principianti che, volendo bruciare le tappe della conoscenza del proprio computer, tendono ad esclamare: "Vorrei, ma... posso" (grazie, ovviamente, anche alla nostra rivista ed all'altrettanto nostra immodestia).

A proposito: vi siete accorti che, nel primo file batch, è possibile usare con successo l'utility Whereis.com?

Pensate, allora, a come agire per individuare il file da "trattare" con Type; e non dimenticate di farcelo sapere, dimostrando di aver raccolto la "sfida" lanciata in queste stesse pagine...

Aiutare i bisognosi

La nostra rivista si rivolge a tutti utenti, appassionati di informatica, e quindi anche a coloro che posseggono un **computer Ms-Dos** compatibile.

Tra questi è inevitabile che vi siano coloro che, per la prima volta, hanno a che fare con una tastiera. Nulla di male, quindi, se vengono affrontati argomenti semplici, e descritti senza avarizia di parole, per evitare pericolosi scoraggiamenti dopo i primi, inevitabili, insuccessi.

Se è vero, infatti, che basta digitare un carattere in più, o in meno, per far visualizzare messaggi di errore, è anche vero che a volte, pur rispettando in pieno le regole sintattiche, non si ottiene ciò che si sperava; a volte ciò succede perchè il computer non è stato correttamente inizializzato, a volte perchè non è presente, in memoria, un certo programma, a volte... e chi lo sa!

Ciò che il nostro lettore troverà in queste pagine non sarà, quindi, una mera descrizione delle potenzialità del sistema Ms-Dos.

E' nostra presunzione, invece, affrontare i vari argomenti, e sono tanti, stimolando il lettore nella ricerca e sperimentazione personale, adattando al proprio elaboratore le nozioni apprese.

Più che altro vogliamo portare avanti una metodologia che possa sempre essere di aiuto per il lettore che, in più di un'occasione, rischia di impantanarsi in procedure errate, apparententemente insolubili. Nonostante gli argomenti affrontati, grazie alla semplicità che li contraddistingue, siano applicabili a qualsiasi compatibile Ms-Dos, precisiamo (ma solo per dovere di cronaca) che i vari "esperimenti" descritti sono stati compiuti su un elaboratore AT compatibile dotato di scheda hercules e monitor b/n; due drive per minifloppy (1.2 Mbyte e 360K) ed un Hdisk da 20 Mega completano la dotazione.

di Alessandro de Simone

A PROPOSITO DI STAMPANTI...

Chi deve acquistare una nuova stampante si perde nella marea di modelli offerti. Ma è proprio così difficile scegliere una stampante?

Chi è abituato alle stampanti della precedente generazione ritiene che una stampante sia soltanto un accessorio "passivo" da collegare al computer più o meno come facciamo con la tastiera.

Oggi, invece, una stampante è un vero e proprio elaboratore, con tanto di micprocessore, programma di gestione su Eprom, memoria Ram. Nei tempi antichi la periferica era totalmente gestita dal computer, con conseguente perdita di tempo in fase di elaborazione dati. In seguito venne introdotto il cosiddetto **buffer** di memoria Ram, che consentiva al computer di scaricare il testo da stampare in modo da esser libero per successive elaborazioni. Per farla breve, anche le stampanti più economiche offrono, all'utente finale, potenzialità prima inimmaginabili.

Il prezzo di vendita al pubblico di una certa tipologia di apparecchi, di solito, risulta decisamente allineato con periferiche di analoghe caratteristiche.

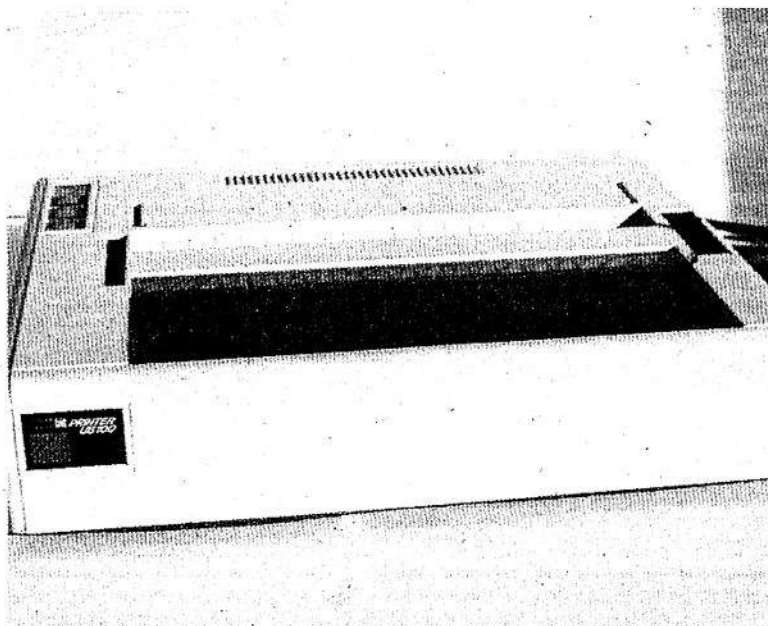
Quale stampante, allora, conviene acquistare?

La risposta, ovviamente, non può essere univoca, ma cercheremo egualmente di aiutarvi a trovarla.

Come per la scelta di un computer, è opportuno riflettere su ciò che **effettivamente** serve per soddisfare le nostre esigenze. E' inutile, ad esempio, sognare una stampante laser se la "produzione" mensile media è di poche decine di pagine: risulta decisamente più economico lavorare con una modesta stampante ad aghi e ricorrere ai servizi di una copisteria quando occorre una riproduzione al laser.

Al contrario, la decisione di presentare una (magari voluminosa) tesi di laurea con qualità laser potrebbe, a conti fatti, privilegiare l'acquisto di un simile apparecchio.

La velocità di stampa è un'altra caratteristica spesso **troppo** venerata. Un hobbysta, che si limita a stampar listati o semplici testi e tabelle, non dovrebbe decidere l'acquisto della periferica in base a tale caratteristica.

A che serve ottenere un listato su carta in due minuti contro il minuto e 30 secondi vantato da un modello concorrente? La

```
*TIPO3 (27,101,83,1,3)
*TIPO5 (27,101,83,1,5)
*TIPO7 (27,101,83,1,7)
*Formato 1 x 1 (27,117,3,48,49,48,49)
*Fine caratteri evidenziati (27,101,83,0,0)
*Caratteri compressi (27,15)
*Caratteri comp. Off (18)
*Caratteri esponenti (27,83,48)
*Caratteri deponenti (27,83,49)
*Esp + Dep. Off (27,84)
*Sottolineat. singola (27,101,85,0,27,45,1)
*Sottol. sing. neretto (27,101,85,1,27,45,1)
*Sot. sin. ner. marcato (27,101,85,2,27,45,1)
*Sottolineatura doppia (27,101,85,3,27,45,1)
*Sottol. doppia neretto (27,101,85,4,27,45,1)
*Sot. dop. ner. marcato (27,101,85,5,27,45,1)
*Sottolineatura off (27,45,0)
*Sopralineatura attivata (27,101,111,1)
*Sopralineatura Off (27,101,111,48)
*Fondino tipo 1 (27,101,73,1,27,101,76,49)
*Fondino tipo 2 (27,101,73,2,27,101,76,49)
*Fondino tipo 4 (27,101,73,4,27,101,76,49)
[illegible]
*Disattiva Fondino (27,101,73,1,27,101,76,48)
*Doppia lar. (27,14,1)
*Doppia larghezza (1 rigo) Off (20)
*Doppia lar. (27,87,1)
*Doppia largh. Off (27,87,0)
*Caratteri in doppia altezza (27,86,1)
*Caratteri in altezza normale (27,86,0)
*Carattere corsivo (27,52)
*Carattere corsivo stop (27,53)
*Caratteri in neretto (27,71)
*Fine caratteri in neretto (27,72)
*Stampa ombreggiata (27,69)
*Stampa ombreggiata off (27,70)
*Margine a 80 caratteri (27,81,80)
```

velocità di stampa, quindi, deve interessare solo chi ha l'esigenza di un elevato "volume" di stampa quotidiana (studi professionali, redazioni di riviste, stazioni di word processing in generale).

Anche la possibilità di effettuare più copie contemporaneamente viene valutato, a nostro parere, con troppa rilevanza.

Capita molto spesso di effettuare quadruple copie? E in quei rari casi in cui ciò dovesse verificarsi, risulta così sconvolgente effettuare le quattro copie, l'una dopo l'altra? E' ovvio che in alcuni casi particolari (studi di ragioneria e commercialisti in genere) la possibilità di stampare più copie in contemporanea (pensate ai moduli delle tasse...) è un'esigenza irrinunciabile.

Allo stesso modo, non deve interessare la possibilità di emulare una dozzina di **protocolli** quando tutti i programmi, prevalentemente usati dall'utente, consentono almeno una delle (pur se poche) emulazioni supportate dal modello che stiamo per acquistare.

I parametri da valutare al momento dell'acquisto, pertanto, devono limitarsi allo stretto indispensabile.

Se, poi, un certo modello offre potenzialità maggiori, tanto meglio.

La possibilità di usare un **nastro a colori** deve essere valutata con attenzione, non tanto a causa del prezzo del nastro di ricambio (ormai modesto) ma anche per l'effettiva utilità di un simile optional. La possibilità di usare fogli di formato maggiore dell'**A-4** deve interessare solo in casi particolari

Inutile ricordare che una stampante riproduce, così come le vengono inviati, i vari caratteri; a meno che uno di questi non sia il carattere "speciale" **ESC Chr$(27)**. Questo carattere, in pratica, comunica alla stampante un messaggio del genere:

"Il prossimo carattere che ricevi non rappresenta un carattere da stampare, ma un comando da eseguire secondo il programma presente nella tua Eprom".

E' ovvio che la stampante, al suo interno, ha un "decodificatore" in grado di eseguire i vari ordini impartiti grazie ad Esc.

E' in grado, ad esempio, di individuare il numero di caratteri da considerare come codici e di separarli da quelli di testo.

I listati di queste pagine, e l'effetto che generano, serviranno meglio a chiarire il

```
*Reset printer (27,64)
*Margine a 45 caratteri (27,81,45)
*Carattere normale (27,101,83,0,0)
*Carattere 3 x 2 (27,117,3,48,52,48,51)
*Carattere 2 x 3 (27,117,3,48,51,48,52)
*Altezza normale (27,117,3,48,49,48,49)
*Carattere normale (27,101,83,0,0)
*Carattere ombreggiato (27,101,83,2,0)
*Carattere ombregg. evidenz. (27,101,83,3,0)
*Caratt. sottolineat. sottile (27,101,83,4,0)
*Caratt. leggera ombregg. (27,101,83,5,0)
*Car. legg.ombr + sott. sott. (27,101,83,6,0)
*Carattere evidenziato (27,101,83,1,0)
```

modo con cui si può programmare una stampante.

Di solito i vari codici possono essere inviati da qualsiasi computer (**Amiga** ed **Ms-Dos** soprattutto) in qualsiasi linguaggio di programmazione e, se lo desiderate, anche mediante opportune selezioni con il pannello di comandi presente in una qualsiasi stampante.

Che cosa stampare

L'utente principiante può spaventarsi al pensiero di dover settare opportunamente una stampante prima di usarla. In pratica, però, vi possiamo assicurare che è sufficiente collegare la periferica, appena acquistata, al computer ed accendere i due apparecchi.

Di solito le stampanti sono settate in modo da essere immediatamente usate con i programmi professionali più diffusi. Un testo redatto con **Word Star** (ed una tabella preparata con **Quattro**) vengono riprodotte senza alcun problema di settaggio particolare.

Perfino l'**hard copy** di una schermata grafica (realizzabile, ad esempio, con **DigiPaint II**) può esser "sparata" fuori con la massima disinvoltura (in un tempo che, però, supera spesso i 10 minuti...).

Non tutti sanno che, una volta inviato (mediante opportuna sequenza di **ESC**) il comando che imposta una certa modalità di stampa (**corsivo, neretto, doppia altezza, font di carattere** e così via), questa rimane attiva fino a nuovo ordine.

E' quindi possibile creare una serie di files che, con la massima semplicità, siano in grado di settare la stampante secondo una modalità ben precisa.

Supponiamo, ad esempio, di voler stampare le directory dei nostri dischetti (o altri files di testo, come i listati di programmi) secondo una modalità di nostro gradimento, come ad esempio quella indicata come **ombreggiata ed evidenziata** sulla nostra ipotetica stampante (ogni modello può offrire sequenze diverse di codici Esc, ma il concetto non cambia).

Per essere sicuri di selezionare solo questa opzione (e di eliminarne altre, eventualmente già presenti) è indispensabile, dapprima, inviare anche il comando di **Reset** (che supponiamo sia **Esc @**

```
Esempio di stampa di codici a barre
con la stampante FUJITSU DL-1110
Tempo di stampa di 3 codici = 26 secondi
```

```
LPRINT "Esempio di stampa di codici a barre"
LPRINT "con la stampante FUJITSU DL-1110"
LPRINT "Tempo di stampa di 3 codici = 26 secondi"
LPRINT : CLS
Stringa$ = "123456789012": Lunghezza = LEN(Stringa$) + 6
Alt = 1440: Altezza = INT(Alt / 24)
LPRINT CHR$(27); CHR$(64): REM Reset printer
FOR i = 1 TO 3: LPRINT "A"; "  "; CHR$(27); CHR$(20);
LPRINT CHR$(Lunghezza); CHR$(82); CHR$(50); CHR$(24);
LPRINT CHR$(Altezza); CHR$(1); Stringa$; "  "; "  "; "B";
NEXT: LPRINT
```

```
REM Tempo impiegato: min. 2 sec. 23
CLS : LPRINT CHR$(27); CHR$(86); CHR$(1);: ' reset
LPRINT "Modi di stampa"
LPRINT : LPRINT : CMD$ = "": WHILE a$ <> "*"
READ a$: IF a$ = "*" THEN END
IF LEFT$(a$, 1) <> "*" THEN
LPRINT CHR$(VAL(a$)); : CMD$ = CMD$ + a$ + ","
ELSE LPRINT a$; " ("; LEFT$(CMD$, LEN(CMD$) - 1); ")": CMD$ = ""
END IF
WEND
DATA 27, 64, *Reset printer
DATA 27, 81, 45, *Margine a 45 caratteri
DATA 27, 101, 83, 0, 0, *Carattere normale
DATA 27, 117, 3, 48, 52, 48, 51, *Carattere 3 x 2
DATA 27, 117, 3, 48, 51, 48, 52, *Carattere 2 x 3
DATA 27, 117, 3, 48, 49, 48, 49, *Altezza normale
DATA 27, 101, 83, 0, 0, *Carattere normale
DATA 27, 101, 83, 2, 0, *Carattere ombreggiato
DATA 27, 101, 83, 3, 0, *Carattere ombregg. evidenz.
DATA 27, 101, 83, 1, 0, *Carattere evidenziato
DATA 27, 117, 3, 48, 50, 48, 50, *Matrice 2 x 2
DATA 27, 101, 83, 1, 0, *TIPO1
DATA 27, 117, 3, 48, 49, 48, 49, *Formato 1 x 1
DATA 27, 101, 83, 0, 0, *Fine caratteri evidenziati
DATA 27, 15, *Caratteri compressi
```

Tutte le istruzioni **DATA** che *sembrano* mancare nel presente listato, (omesse per esigenze di spazio) possono esser desunte dall'output di stampa riportato in queste stesse pagine. Il "trucco" consiste, infatti, nel trascrivere i vari codici (in decimale), separati da una virgola; l'ultimo DATA di ciascun rigo, che contiene un commento, **deve** esser preceduto da un asterisco. Lo schema del listato (valido per **Amiga** ed **Ms-Dos**) può essere usato per evidenziare le caratteristiche di stampa di qualsiasi stampante. A patto, ovviamente, di inserire la giusta sequenza di codici.

```
DATA 27, 101, 85, 0, 27, 45, 1, *Sottolineat. singola
DATA 27, 101, 85, 1, 27, 45, 1, *Sottol. sing. neretto
DATA 27, 101, 85, 2, 27, 45, 1, *Sot. sin. ner. marcato
DATA 27, 101, 85, 3, 27, 45, 1, *Sottolineatura doppia
DATA 27, 101, 85, 4, 27, 45, 1, *Sottol. doppia neretto
DATA 27, 101, 85, 5, 27, 45, 1, *Sot. dop. ner. marcato
DATA 27, 101, 111, 1, *Sopralineatura attivata
DATA 27, 101, 73, 1, 27, 101, 76, 49, *Fondino tipo 1
DATA 27, 101, 73, 1, 27, 101, 76, 48, *Disattiva Fondino
DATA 27, 86, 1, *Caratteri in doppia altezza
DATA 27, 52, *Carattere corsivo
DATA 27, 71, *Caratteri in neretto
DATA 27, 69, *Stampa ombreggiata
DATA 27, 81, 80, *Margine a 80 caratteri
DATA *: REM Attenzione! l'ultimo DATA deve essere un solo asterisco.
END
```

Listato di prova (AmigaBasic oppure Quick Basic)

Ecco il listato che genera l'output visibile in queste pagine.
Per ogni stampante, ad ogni sequenza di codici corrisponde una modalità di stampa ben precisa.

Esc). Non sempre è possibile "scrivere", con un text editor, caratteri speciali come Esc. Ne consegue che siamo obbligati ad usare un linguaggio (il Basic va benissimo) oppure, come qui suggeriamo, uno degli innumerevoli Tools di utilità.

Operando con un computer Ms-Dos e con **PCTools**, ad esempio, create un file di testo (opzione **W**) di nome **Evidenz** e "riempitelo" di una dozzina di vocali "**a**". Una volta registrato, "entriamoci" con il comando **E**(dit) e, con le opzioni **F1** (toggle mode) ed **F3**, trasformiamo le "**a**" con i comandi necessari.

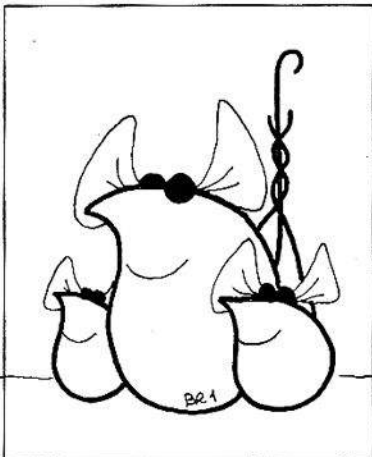
Ciò significa che, sul primo rigo, deve esser presente la sequenza di codici **1B, 40, 1B** (cioè il reset della stampante); subito dopo: **1B, 75, 33** (controllo interlinea automatizzato); poi: **30, 31** (ingrandimento asse x) **30, 31** (ingrandimento asse Y) **1B, 65, 53** (attivazione selezione stile) **31** (carattere evidenziato) **37** (tipo di pattern scuro). Il file viene poi completato con **0A, 0D** (ritorno carrello) ed i rimanenti caratteri "**a**" trasformati in innocui spazi bianchi (20).

Insomma, la schermata di PCTools, relativa al file di nome Evidenz, dovrà apparire come in figura, prima di impartire il comando di salvataggio **F5** (update).

A questo punto, tutte le volte che invieremo il file **Evidenz** alla stampante, con...

Evidenz > prn

...questa varrà settata come indicato, e stamperà, tutto ciò che invieremo, secondo la modalità selezionata.

L'effetto che ne risulta si può osservare in queste stesse pagine.

La stampante: piccolo glossario per chi è digiuno di termini tecnici

Attributi del carattere. Sono le caratteristiche del **font**, lievemente modificate. In parole povere, indicano la possibilità di effettuare sottolineature, ottenere caratteri in **neretto**, in *corsivo* e così via. Di solito è possibile ottenere più attributi contemporaneamente (esempio: ***carattere corsivo, neretto e sottolineato***).

Buffer. Indica la capacità della stampante di ricevere, ad alta velocità, un certo numero di caratteri, da stampare un po' per volta. Più capiente è il buffer, più velocemente il computer si "libera" del testo da stampare in modo da esser disponibile per svolgere altre elaborazioni. Non sempre un buffer molto capiente rappresenta un vantaggio. Se, ad esempio, viene inviato (per errore) un lungo documento da stampare, la riproduzione su carta continua inesorabile fino all'ultimo carattere, a meno che non si invii un carattere di svuotamento o si spegne l'apparecchio. Per evitare questi errori è possibile selezionare il buffer in modo da consentire la stampa di una quantità modesta di caratteri. E' ovvio, però, che in questo modo il computer risulta "bloccato" finchè il testo non viene inviato per intero, buffer dopo buffer.

Carattere. E' la "forma" dei caratteri da stampare; chi ha utilizzato una macchina da scrivere a margherita (o a sfera), sa benissimo che, a seconda della margherita o della sfera innestata, ottiene altrettanti **font**. Con una stampante ad aghi è possibile programmare i vari alfabeti ed ottenere una stampa con caratteri diversi, anche su di uno stesso rigo, semplicemente mediante comandi impartibili via software. Il font non deve esser confonfuso con gli attributi del carattere (vedi). Esempi di Font sono: `Courier`, `Prestige`, `Orator` e così via.

Cella carattere. E' la matrice di punti in cui viene "racchiuso" un carattere del set selezionato. La qualità di stampa aumenta al crescere delle dimensioni della matrice; la velocità di stampa, invece, decresce.

Cpi: caratteri per inch (= pollice). Indica il numero di caratteri che è possibile stampare in un pollice (2.54 mm.) e fornisce un'idea, in pratica, della "compressione" del carattere. Più caratteri riescono ad essere ospitati in un pollice, minore è la loro larghezza. Un elevato valore del cpi permette di stampare, in particolare, tabelle dotate di numerose colonne; un valore alto di cpi offre una migliore leggibilità del testo.

Cps. Caratteri per secondo. Indica la velocità di stampa espressa come numero di caratteri riproducibili al secondo. La velocità varia enormemente a seconda della qualità (vedi) di stampa desiderata. E' ovvio che una qualità inferiore consente alla testina di scrittura di effettuare una sola passata.

Di solito un documento viene dapprima stampato ad alta velocità (e bassa qualità); solo quando sono state effettuate eventuali correzioni conviene selezionare una migliore qualità di stampa, pur se al prezzo di una minore velocità.

La stampa ad alta velocità, inoltre, presenta il vantaggio di limitare il decadimento del nastro inchiostrato.

Emulazione. E' la capacità di un apparecchio di offrire gli stessi risultati ottenibili dall'apparecchio che emula. Ad esempio, affermare che una certa stampante emula il protocollo **Epson LQ-2500**, significa che quella stampante presenterà, su carta, lo stesso output che si sarebbe ottenuto utilizzando una stampante originale Epsòn della serie LQ-2500.

Interfaccia. E' il sistema di "comunicazione" tra computer e stampante. Il più diffuso e quello **Centronics**, tant'è vero che tutti i computer offrono, di serie, questo tipo di interfaccia.

Alcuni programmi, o procedure di stampa, possono richiedere l'interfaccia **RS-232**, a volte più lenta della Centronics.

Pitch carattere. Indica il numero di caratteri per pollice stampabili (vedi cpi).

```
   PC Tools Deluxe R4.22                                       Vol Label=DOS400
-------------------------------File View/Edit Service----------------------------
Path=C:\*.*
File=EVIDENZ.        Relative sector 0000000, Clust 06206, Disk Abs Sec 0025013
Displacement ----------------- Hex codes--------------------     ASCII value
0000(0000)   1B 40 1B 1B 75 33 30 31 30 31 1B 65 53 31 37 0A    @  u30101 eS17
0016(0010)   0D 20 20 1A 00 00 00 00 00 00 00 00 00 00 00 00
0032(0020)   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0048(0030)   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0064(0040)   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0080(0050)   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0096(0060)   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0112(0070)   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0128(0080)   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0144(0090)   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0160(00A0)   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0176(00B0)   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0192(00C0)   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0208(00D0)   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0224(00E0)   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Uso di PCTools

*Una sequenza di **ESC**ape può essere memorizzata stabilmente su un file da "redirigere" verso la stampante tutte le volte che lo vogliamo. Il file di esempio, di nome Evidenz (citato nel testo) contiene ben **17** caratteri / codice. Tutte le volte che viene inviato alla stampante, questa stamperà i testi secondo l'impostazione impartita.*

Ad esempio con...

Type Testo > prn

*...il file di nome **Testo** apparirà su carta con caratteri aventi la caratteristica impostata.*

di Giancarlo Mariani

ASSEMBLER 80X86, PRIMI PASSI

"I primi cinque minuti sono sempre i più difficili", afferma una nota campagna pubblicitaria. Ciò vale anche per il linguaggio Assembler dei computer Ms-Dos compatibili. Vediamo, allora, di metterci all'opera senza problemi

I compatibili PC (ma è meglio chiamarli **Ms-Dos** compatibili) si sono ormai diffusi a macchia d'olio invadendo perfino il mercato degli home computer. Comprare oggi un PC in configurazione completa non costa molto di più di quanto costava, 4 anni fa, un "equivalente" sistema basato sul **C/64**.

Ovviamente il PC è più adatto per scopi professionali dal momento che un microprocessore **80286** a 16Mhz, un disco fisso da **40 Mb**, una scheda grafica EGA (meglio se **VGA**), **1MB** di memoria ed una **stampante** sufficientemente veloce costituiscono un mixer adatto alla maggior parte delle esigenze, ad un prezzo decisamente abbordabile. Per questa ragione, ma anche per la enorme diffusione di software, moltissime s/w house, tra cui le famose **Borland** e **Microsoft**, hanno sviluppato linguaggi di programmazione adatti ad ogni scopo. Non esiste un linguaggio di programmazione che non sia stato implementato sul PC, a cominciare dal "vecchio" **GWBASIC**, per passare ai più moderni **QuickBasic** o Turbo-Basic, agli "strutturati" **Pascal** e **C**, al didattico **Logo**, all'AUTOCADdiano **Lisp**, e poi **Prolog, Ada, Fortran, Cobol**, e, perchè no?, anche il "caro e vecchio" **Assembler**.

L'Assembler, vecchio ricordo di smanettoni 64isti, animale quasi dimenticato sul PC, risulta invece praticamente indispensabile per applicazioni particolari in cui necessita una velocità di esecuzione particolarmente elevata; oppure un'interazione diretta con la macchina (cosa tra l'altro possibile anche con altri linguaggi ad alto livello).

Sebbene alcuni di questi ultimi, come ad esempio il **C**, permettano di ottenere una velocità di esecuzione spesso di poco inferiore a quella del linguaggio macchina (**LM**) "puro", non possono essere usati convenientemente per applicazioni nelle quali l'altissima velocità di esecuzione è un parametro fondamentale per il buon funzionamento del programma.

Pensate forse che alcune software-house avrebbero potuto dichiarare (a ragione!) che il loro compilatore C compila 5000 linee al minuto, se non l'avessero sviluppato in LM?

L'apprendimento dell'Assembler risulta comunque utile per vari motivi.

Anzitutto, permette di conoscere più a fondo la macchina con la quale si opera, dal momento che **il LM interagisce direttamente** sia con il microprocessore sia con l'hardware del computer. In secondo luogo può costituire un valido esercizio sia per individuare tecniche di programmazione più semplici e veloci (esperienza che si rivelarà utile anche con i linguaggi ad alto livello) sia... per ricordare i vecchi tempi, quando perdevamo pomeriggi interi sul C/64 nel tentare la sprotezione di qualche programma navigando tra fiumi di listati Assembler e locazioni di memoria.

La precedente conoscenza di altri linguaggi, ed in particolar modo di **altri Assembler** (tra cui quello specifico dei **6502**), facilitano enormemente l'apprendimento dell'Assembler serie **80XXX** (cioè 8088, 8086, 80286, 80386, 80486) perchè, fondamentalmente, **le tecniche di programmazione sono sempre le stesse.**

Le maggiori **differenze** riguardano il maggior numero di istruzioni a disposizione (ma questa è una facilitazione, almeno rispetto al C/64) ed una gestione della memoria particolare, ossia **a segmenti**, che può disorientare il principiante.

Un computer Ms-Dos può indirizzare N-megabytes di memoria, ma dato che il bus degli indirizzi ha un numero di **pin** (=piedini) limitato, la memoria stessa viene suddivisa in blocchi da **64 kbytes** ciascuno, chiamati **segmenti**. E' questa una caratteristica che discuteremo tra breve e che, anche se complica leggermente la programmazione, può portare notevoli vantaggi.

I processori del PC

Prima di iniziare un discorso più corposo, è opportuno fare una breve

introduzione sul nero millepiedi che dà vita al nostro PC compatibile.

Il microprocessore in questione appartiene alla famiglia 80XXX; i primi esemplari avevano il nome di **8088** ed operavano a 4.77 Mhz.

L'evoluzione ha portato allo sviluppo dell'8086, 80186, 80286, 80386, 80486, 80... (gli altri li vedremo in un prossimo futuro).

Le differenze principali tra i vari processori risiedono nel **numero** di istruzioni LM a disposizione e, ovviamente, nella **velocità** di esecuzione delle stesse. Un 80386, anche a parità di clock, risulta molto più veloce di un 8086. I processori dispongono di un set di istruzioni compatibili tra di loro **verso il basso**. Ciò vuol dire che un programma sviluppato per un 8088 girerà tranquillamente su tutti i successivi processori. Un programma sviluppato per 80386, invece, girerà sugli altri processori solo se saranno state utilizzate le istruzioni comuni a tutti, e non soltanto quelle specifiche dell'80386.

In ogni caso, un processore "superiore" sarà **sempre** in grado di interpretare tutte le istruzioni di quelli inferiori, oltre alle istruzioni o accorgimenti particolari che permettono di velocizzare l'esecuzione.

Per fare un paragone banale, una qualsiasi versione di linguaggio Basic sarà sempre in grado di interpretare l'istruzione **Print**, mentre le vecchie versioni non potranno interpretare un'istruzione più "recente", come **Plot**.

Non a caso il famoso programma (che, poi, è un vero e proprio **sistema operativo**) "Windows" è disponibile in più versioni; una specifica per l'8086, che "gira" però anche sui sistemi superiori; altre, specifiche per sistemi basati su 80286 e 80386 (che utilizzano le nuove istruzioni di quei processori) sono in grado di offrire nuove potenzialità.

In queste pagine parleremo, almeno all'inizio, del processore più... anziano (ossia l'8086) in modo che tutti i possessori di PC, dai vecchi XT ai nuovissimi sistemi 80486, possano provare i programmi proposti.

In seguito, esaminando le istruzioni dei processori "superiori", l'esperienza accumulata sarà comunque indispensabile per apprendere le nuove potenzialità offerte dai computer delle nuove generazioni.

Dentro il processore

Vediamo ora come è strutturato un microprocessore.

L'8086 possiede, al suo interno, ben **14 registri** a 16 bit, che possono essere raggruppati in registri **generali**, registri di **controllo** e registri di **segmento**.

I registri generali possono essere, a loro volta, suddivisi in registri per **dati**, registri **indice** e registri **puntatore**. Per esaminarli in dettaglio, riferitevi agli specifici riquadri.

Indirizzi in memoria

La memoria di un PC è organizzata in un insieme di **segmenti**, ciascuno dei quali può essere lungo fino a **64 kbytes**.

Gli indirizzi assoluti in un PC, però, variano dal valore esadecimale **00000** fino a **FFFFF**; ne consegue che 16 bit non sono sufficienti a contenere l'intero indirizzo dal momento che, con 16 bit, è possibile arrivare "solo" fino al valore FFFF.

Ecco, quindi, spiegato la funzione svolta dalla suddivisione in segmenti.

Ogni indirizzo assoluto è formato da una **base** a 16 bit (destinata a specificare l'indirizzo di **inizio** del segmento), più un **offset**, anch'esso a 16 bit, che specifica l'indirizzo all'**interno** del segmento stesso.

La base, ossia il segmento, è contenuto in uno dei 4 registri, appunto, di segmento **CS, SS, DS** oppure **ES**.

L'indirizzo assoluto viene calcolato SHIFTando la base di 4 bit a sinistra e sommando l'offset al valore così calcolato.

I registri per i dati

Sono 4 registri a 16 bit, chiamati **AX, BX, CX** e **DX**. Sono costituiti, ciascuno, da due gruppi di 8 bit, indirizzabili separatamente; ciò significa che i registri possono essere usati indifferentemente a 16 oppure ad 8 bit. Per operazioni a 8 bit i registri si dividono in parte **bassa** (Hi) e parte **alta** (Lo), in accordo con la seguente tabella:

16 bit	**8bit** *(HI)*	*(LO)*
AX	AH	AL
BX	BH	BL
CX	CH	CL
DX	DH	DL

I registri possono essere usati indifferentemente per operazioni aritmetiche o logiche; alcuni di essi possono svolgere anche altre funzioni particolari. **CX**, ad esempio, è frequentemente usato come contatore, **BX** come registro di base per alcuni modi di indirizzamento.

I registri indice e puntatore

Nell'8086 vi sono due registri puntatore, che servono come **stack pointer** nelle varie operazioni di gestione del flusso di programma. Si tratta di **BP** e **SP**, entrambi a 16 bit. **SP** è il "vero" puntatore allo stack, e funziona come su qualsiasi altro processore, ossia "punta" agli indirizzi di ritorno di subroutine e interrupt memorizzati nello stack. **BP**, invece, può essere usato come copia di SP oppure per creare stack indipendenti nelle subroutines per la memorizzazione dei parametri. I registri indice si chiamano **SI** (=source index) e **DI** (=destination index). Sono anch'essi a 16 bit e si rivelano particolarmente indicati sia nella manipolazione di stringhe, sia nella gestione di alcuni modi di indirizzamento dell'8086.

Entrambi hanno la possibilità di auto incremento o decremento.

I registri SP, BP, SI e DI hanno, inoltre, la possibilità di venire usati come registri per dati (AX, BX, CX, DX) con tutte le istruzioni aritmetiche e logiche a disposizione. Ma di questo ci occuperemo dopo.

I registri di controllo

L'8086 possiede due registri di controllo, denominati **IP** (=instruction pointer) e il registro dei **Flags** o Status Word. IP è simile al Program Counter usato in altri processori e "punta" alla successiva istruzione da eseguire nel programma LM.

Lo status word o Flags è una specie di pro-memoria che contiene gli "effetti" generati dall'ultima operazione aritmetica o logica eseguita.

E' composto da 16 bit, dei quali solo **9** sono utilizzati:

- 0: **CF** Carry flag
- 2: **PF** Parity flag
- 4: **AF** Half carry flag
- 6: **ZF** Zero flag
- 7: **SF** Sign flag
- 8: **TF** Single step flag
- 9: **IF** Interrupt enable flag
- 10: **DF** Direction flag
- 11: **OF** Overflow flag

I flags vengono controllati dalle istruzioni di salto condizionato ed in generale da tutte le istruzioni che controllano una qualche condizione.

Si ottiene, in tal modo, un indirizzo a **20 bit** (16 + 4) che permette di indirizzare fino a **1 MByte** di memoria. Facciamo un esempio: supponiamo di avere una base che vale **1243** (esadecimale, che corrisponde al binario **0001 0010 0100 0011**) ed un offset che vale **1325** (esadecimale, che corrisponde al binario **0001 0011 0010 0101**). Dapprima andrà shiftata la base di 4 bit a sinistra (aggiungendo 4 zeri in coda) ottenendo il valore di **12430**. In seguito andrà sommato l'offset; quindi:

```
1  2430  +
   1325  =
1  3755
```

L'indirizzo esadecimale **13755** è quindi l'indirizzo assoluto. Sia la base che l'offset possono variare liberamente tra **0** e **65535** (**FFFF** esadecimale); può quindi capitare che, combinando opportunamente i due valori, si ottenga lo stesso indirizzo assoluto anche con valori che possono sembrare molto diversi tra loro. Particolare attenzione deve quindi esser prestata alla corretta inizializzazione dei segmenti; in caso contrario potrebbero verificarsi imprevedibili (e sgradevolissime...) sovrascritture di dati all'interno dell'area programma.

Ad esempio, con una base che vale **1141** ed un offset di **2345**, si ottiene esattamente l'indirizzo assoluto precedentemente esaminato:

```
1  1410  +
   2345  =
1  3755  (!!)
```

Segmenti

Dei 4 registri di segmento (denominati **CS, DS, SS, ES**), il primo, ossia il **Code Segment**, contiene la base che servirà per calcolare l'indirizzo assoluto dove è contenuto il codice, ossia il programma LM, che deve essere eseguito dal processore.

Prima di eseguire una qualsiasi istruzione, l'8086 farà riferimento a **CS** che, shiftato di 4 bit a sinistra (e sommato a **IP**, l'instruction **p**ointer) fornirà l'indirizzo assoluto dell'istruzione da eseguire.

Il registro **DS** (= **D**ata **S**egment) viene usato come base per il segmento che dovrà contenere i **dati** del programma. In pratica, quasi ogni istruzione che accede alla memoria, allo scopo di leggere o scrivere dati, prenderà come riferimento DS, che verrà shiftato e sommato all'indirizzo assoluto specificato nell'istruzione stessa.

Ad esempio, l'istruzione...

MOV AH, 3000h

...che pone nel registro **AH** il contenuto della locazione di memoria *relativa* 3000h, non leggerà il contenuto del "semplice" indirizzo 3000h, ma quello risultante dalla **somma** di 3000h + DS shiftato di 4 bit a sinistra.

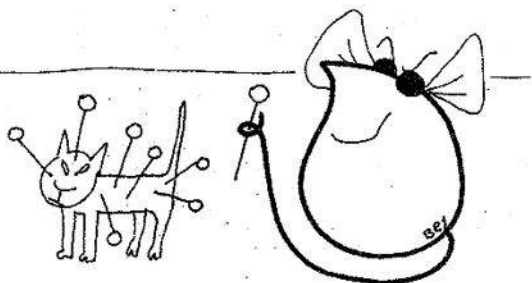
Il registro **SS** (=**S**tack **S**egment) serve come base per lo stack. Quando si "invoca" una subroutine o si salva un qualche valore nello stack, il salvataggio verrà effettuato dall'8086 prendendo in considerazione l'indirizzo contenuto nel registro **SP** (**S**tack **P**ointer) **sommato** a SS shiftato di 4 bit.

Il registro **ES** (=**E**xtra **S**egment) viene usato in alcune istruzioni per indirizzare

I registri di segmento

Quattro registri speciali a 16 bit, cioè **CS** (code segment), **SS** (Stack segment), **DS** (Data segment) e **ES** (Extra segment) sono usati dall'8086 per la determinazione degli indirizzi assoluti di memoria, e definiscono i segmenti, rispettivamente, per il codice, per lo stack, per i dati e un segmento extra a scopo generale.

14 Registri (16 BITS)		
	Generali	Dati (AX, BX, CX, DX)
		Indice (SI, DI)
		Puntatore (BP, SP)
	Controllo	IP, SW
	Segmento	CS, SS, DS, ES

Il registri dei microprocessore 8086. Pur se "fisicamente" formati da 16 bits, possono esser considerati come due gruppi da 8 bit ciascuno.

dati (in modo simile a DS), ma può essere manovrato abbastanza liberamente dall'utente e può puntare indifferentemente ad un segmento contenente codice o dati.

Dal momento che all'interno di un qualsiasi segmento gli indirizzi variano sempre tra 0 e 65535, il cambio del segmento non influisce sul valore degli indirizzi (o meglio, offset) contenuti nel programma LM.

Ciò significa che **i programmi LM sono sempre e completamente rilocabili**, e possono funzionare in qualsiasi area di memoria, semplicemente cambiando il valore di CS e non modificando affatto gli altri indirizzi eventualmente contenuti nel programma.

Questa straordinaria particolarità viene sfruttata a fondo dal sistema Ms-Dos perchè, quando un programma viene caricato da disco, il sistema rintraccia il primo spazio libero in memoria Ram, e vi alloca il programma.

Gli Interrupts

L'8086 possiede due linee di interrupt: una **Mascherabile** (denominata **INTR= interrupt**) ed una **Non mascherabile** (**NMI= n**ot **m**askable **i**nterrupt). Gli interrupts, come su altre macchine, sono normalmente collegati a dispositivi esterni e possono servire per numerosi scopi, quali la gestione della tastiera, del video, di altre schede e così via.

L'interrupt non mascherabile può essere usato per segnalare alla CPU eventi importanti in modo che il processore possa provvedere alle opportune operazioni. I processori superiori (80286...) possiedono più linee di interrupt mascherabile.

Oltre agli interrupt **esterni**, l'8086 possiede anche 256 interrupt **interni**. Si tratta di particolari istruzioni che consentono di "saltare" ad eseguire programmi LM ubicati a partire da locazioni di memoria definite in una tabella (che può contenere fino a 256 indirizzi).

Alcuni di questi interrupts sono usati dal **Bios** ("programma" LM memorizzato su Eprom che serve per inizializzare il sistema al momento dell'accensione del computer) e dallo stesso Dos, che se ne servono come puntatori di ingresso alle proprie funzioni. Anche le istruzioni che manipolano gli interrupts verranno descritte durante le future chiacchierate sull'Assembler.

Macro Assembler Microsoft

Per impadronirsi correttamente dei concetti tipici di un linguaggio di programmazione, è di grande aiuto sviluppare semplici programmi di esercizio.

Se i programmi, poi, sono sviluppati direttamente sul calcolatore, risulta ancora più facile comprenderli, dato che se ne possono osservare immediatamente i risultati.

I programmi LM che descriveremo potrebbero essere introdotti anche da **Basic** con righe di tipo **Read / Data**, come facevamo con il C/64. Con un Ms-Dos compatibile, tuttavia, è molto più semplice utilizzare uno specifico programma assemblatore, in grado di tradurre istruzioni Assembler in codici LM eseguibili.

Il programma assemblatore cui ci riferiremo sarà, inizialmente, uno dei più noti disponibili sul mercato, vale a dire il **Macro Assembler Microsoft** (più conosciuto come **Masm**, versione 5.1). Un qualsiasi altro Assembler servirà egualmente bene allo scopo.

Il **Masm** viene offerto in una elegante confezione comprendente numerosi manuali e 5 dischi. Per seguirci nei nostri esempi, tuttavia, non vi sarà bisogno di tutti i files contenuti nei dischi, ma saranno sufficienti i seguenti:

MASM.EXE (Prog. Macro Assembler)
LINK.EXE (Programma Linker)

Inoltre è consigliabile procurarsi il **Debugger**, composto dai seguenti files:

CV.HLP Help di CodeView Debugger
CV.EXE Prog. CodeView Debugger

Per finire, è necessario un file, che dovrebbe essere fornito assieme al Dos, utile per convertire i files **.EXE** in **.COM**. Il programma si chiama **EXE2BIN.EXE**.

Vi suggeriamo di inserire i **5** files in una directory dal nome **\MASM** in modo da non trovare difficoltà nel seguire gli esempi che descriveremo.

Nel file AUTOEXEC.BAT vi consigliamo di aggiungere il percorso relativo alla nuova directory (PATH = C:\MASM) per rintracciarla più facilmente. Anche se non possedete il disco rigido vi risulterà facile apportare le dovute modifiche necessarie.

Per un corretto funzionamento di Masm il file **CONFIG.SYS** dovrebbe contenere le seguenti istruzioni:

FILES = 20
BUFFERS = 20

Ed è tutto, Siamo (quasi) pronti per iniziare.

I suffissi .EXE e .COM

Nel sistema Ms-Dos vi sono due tipi di programmi che possono essere direttamente "lanciati" ed eseguiti da disco. I files-programma possiedono l'estensione .EXE oppure .COM.

La differenza consiste nel fatto che i files di tipo .COM possono risiedere per intero in un solo segmento; ciò significa che sia il **codice** sia **i dati** sia lo **stack** devono rimanere nel medesimo segmento. Da questo si deduce che un programma di tipo COM **non** può superare la lunghezza totale di 64k, **dati e stack compresi**.

Un programma di tipo EXE, invece, può essere tranquillamente allocato in più segmenti; possiamo quindi avere uno (o più) segmenti di codice, uno (o più) segmenti di dati ed anche di stack.

Un programma EXE può quindi superare, ed anche di molto, il limite dei 64 kbytes.

Per semplificare le cose, almeno all'inizio, faremo riferimento solo a files di tipo .COM, anche se questa scelta comporta la necessità di procurarsi il file EXE2BIN, non compreso nei dischi Masm, ma comunque fornito con moltissime versioni di Dos.

Se non lo possedete, procuratevelo assicurandovi che la versione Dos cui apparteneva sia uguale alla vostra, altrimenti il programma non funzionerà.

Come scrivere in Masm

Un programma in sorgente Assembler deve essere scritto con un qualsiasi editor di testi (si può usare perfino WordStar, ma in modalità "non documento"), ed è composto da una **intestazione**, che impartisce alcune direttive al compilatore, e dalle **righe di programma** vere e proprie, scritte in codice mnemonico Assembler. Il testo sarà poi dato in pasto al Masm, che provvederà a tradurre le istruzioni mnemoniche in codici LM. L'assemblatore ha bisogno di direttive atte a for-

```
;
; Programma di esempio : PROVA.ASM
;
cseg    SEGMENT PARA PUBLIC 'CODE'
        org 100h
        ASSUME cs:cseg, ds:cseg, ss:cseg, es:cseg

Start:
        mov   AX,CS
        mov   DS,AX     ;DS = CS
        mov   ES,AX     ;ES = CS
        mov   AL,'a'    ;Carattere 'a' in AL
        int   29h       ;Scrive il carattere
        mov   AH,4Ch
        mov   AL,0
        int   21h       ;Ritorno al DOS

cseg    ENDS
```

Il tuo primo programma in Assembler 8086!

```
;
; INTESTAZIONE E DIRETTIVE relative a prog. COM
;
cseg    SEGMENT PARA PUBLIC 'CODE'
        org 100h
ASSUME cs:cseg, ds:dseg, ss:cseg, es:cseg
Start:
        mov   AX,CS
        mov   DS,AX     ;DS = CS
        mov   ES,AX     ;ES = CS
        ............
        Qui va scritto il programma assembler
        ............
        mov   AH,4Ch
        mov   AL,0
        int   21h       ;Ritorno al DOS
        ............
        Definizione eventuali dati
        ............
cseg    ENDS
        END   Start
```

nire informazioni sul tipo di programma, sui segmenti, sui dati, ed altre informazioni indispensabili al funzionamento del programma. Per produrre programmi di tipo COM avremo bisogno di un gruppo di istruzioni, contenute nel riquadro *"Intestazioni e direttive"* pubblicato in queste stesse pagine.

Esaminiamo in dettaglio il gruppo di istruzioni:

Le prime righe (che iniziano con il **punto e virgola**) sono semplici **commenti**, e non producono alcun "effetto" sul programma.

Il simbolo di punto e virgola (;) è l'equivalente di **REM** del Basic, e può essere posto in qualsiasi riga in un punto qualunque della stessa.

Tutto ciò che si trova **dopo** il punto e virgola, fino al termine "fisico" della riga, sarà considerato commento, e quindi non considerato ai fini della compilazione.

La riga contenente "cseg SEGMENT" definisce l'inizio di un segmento chiamato appunto **cseg** (il nome è di fantasia; potrebbe essere qualsiasi altro). Il termine **PARA** significa che il segmento deve essere considerato a **para**grafi, ossia a gruppi di **16 bytes**, ma questo non è importante. **PUBLIC** specifica che il segmento è pubblico, ossia, come le variabili definite COMMON del Basic, è "visto" da tutti i moduli che compongono il programma.

La parola **'CODE'** specifica che il segmento dovrà contenere il codice, ossia il programma LM vero e proprio.

Dato che vogliamo produrre un programma di tipo **COM** e che, come detto prima, tali programmi devono essere contenuti interamente in un solo segmento, nel nostro primo programma di esempio non vi saranno altre righe di definizione di segmento.

La riga successiva (**org 100h**) specifica l'indirizzo di origine (relativo all'**interno** del segmento) del programma Assembler. Ciò significa che l'assemblatore, in fase di compilazione, inserirà il codice LM a partire dall'indirizzo **100** esadecimale, relativo al valore di **CS**.

I programmi di tipo EXE non necessitano di una direttiva del tipo ORG, dato che il PC provvede ad allocarli dove meglio crede. I programmi di tipo COM, invece, devono assolutamente iniziare dall'indirizzo 100h.

La riga **ASSUME**.... informa il compilatore che il programma in oggetto dovrà essere di tipo COM.

Infatti, la riga precisa che i 4 registri di segmento (CS, DS, SS, ES) devono essere considerati appartenenti allo stesso segmento (**cseg**).

Le istruzioni **NON** inizializzano effettivamente il valore dei registri di segmento, ma si limitano semplicemente ad informare il compilatore della nostra intenzione di farlo.

"Start:" è un'etichetta che specifica il punto di inizio dell'esecuzione del programma. Potrebbe avere un qualsiasi altro nome tra quelli consentiti.

Le successive tre righe contengono istruzioni Assembler, e servono effettivamente per "sistemare" i registri di segmento.

CS e SS vengono inizializzati automaticamente dal PC all'atto del richiamo del programma, ES e DS invece devono essere inizializzati dall'utente.

Dato che stiamo parlando di programmi COM, i registri DS ed SS dovranno avere lo stesso valore del Code Segment CS.

Per fare ciò viene usata l'istruzione MOV, che "muove" il contenuto di un'area di memoria (o di un registro) in un altro. La prima riga trascrive, nel registro AX, il contenuto di CS; le successive due righe memorizzano in DS ed ES il contenuto di AX e quindi, in pratica, anche di CS. Come si è notato, risulta indispensabile "passare attraverso" un registro di supporto (in questo caso, AX) perchè l'Assembler non consente assegnazioni dirette tra registri di segmento; un'istruzione del tipo **MOV DS,CS**, ad esempio, non è permessa.

Le inizializzazioni sono indispensabili in un programma COM perchè, lo ricordiamo ancora una volta, questo tipo di programma deve essere contenuto in un

segmento ed i 4 registri di segmento devono avere tutti lo stesso valore.

Dopo aver assegnato le prime direttive, si può finalmente scrivere il programma utente Assembler vero e proprio.

Alla fine del programma è necessario ricordarsi di inserire una "chiamata" ad una funzione del Dos (e precisamente la fuzione **4C** richiamata tramite **INT 21h**) che pone termine al programma, rilascia la memoria occupata dallo stesso (in modo da renderla disponibile per altri programmi) e torna al ben noto prompt del Dos **C>** (se, ovviamente, disponete di un disco rigido).

L'istruzione INT può essere considerata come un'istruzione di salto ad una subroutine (in Basic corrisponderebbe a Gosub, e salta ad un indirizzo, posto in una tabella nelle prime pagine di memoria, che punta ad una funzione Dos; in questo caso, appunto, la funzione di fine programma.

I registri AH e AL (quindi, il registro AX) contengono i parametri da passare alla funzione. AH vale **4C** e specifica la sottofunzione da eseguire nella funzione richiamata.

Tramite **INT 21h**, e semplicemente cambiando il valore di AH, è possibile richiamare moltissime funzioni diverse. **AL** (che vale 0) è il codice di ritorno che il programma utente "passa" al Dos quando termina.

A proposito: il codice può essere controllato, da Dos, tramite le istruzioni spesso presenti nei **Batch Files** (avete sicuramente notato, in questo tipo di files, l'istruzione **IF ERRORLEVEL**...).

Generalmente il valore 0 indica che il programma è terminato correttamente, senza evidenziare errori.

Subito dopo l'ultima istruzione in LM si dovranno definire eventuali dati ed aree di memoria usate dal programma, in un modo che vedremo più avanti. E' importante ricordare che un programma COM non può superare la lunghezza di 64k (l'avete capito, speriamo...), quindi la somma del codice e dei dati non potrà superare tale grandezza.

E' da notare anche che il segmento **cseg** è ancora "aperto", quindi effettivamente i dati saranno trascritti proprio in quel segmento, insieme al codice.

I dati, in effetti, potrebbero essere inseriti in punto qualsiasi del programma; in seguito, infatti, il compilatore (o meglio, il **linker**) provvederà a sistemarli al posto che loro compete. Per comodità e chiarezza di lettura dei listati, tuttavia, i dati si inseriscono prima (o dopo) il programma, all'inizio (o alla fine) del testo sorgente. Il nome del segmento aperto, e la parola **ENDS**, determinano la chiusura, cioè la fine, del **segmento**. Nel caso di programmi COM, dato che il segmento è solo uno, queste istruzioni determineranno implicitamente anche la fine del **programma**.

La parola **END** determina inequivocabilmente la fine del programma sorgente, e ciò significa che il compilatore non terrà in considerazione le righe poste **dopo** la parola END.

A destra della parola chiave END deve necessariamente essere digitata la **label** di partenza del programma, cioè il punto in cui il computer dovrà iniziare ad eseguire il programma. Nel nostro caso la label è **Start**, posta all'inizio del programma utente.

Compiliamo i programmi

Descriveremo, ora, la procedura più semplice per compilare ed eseguire i programmi Assembler sorgenti. Il MASM, infatti, possiede numerosissime opzioni, che verranno descritte a mano a mano che sarà necessario.

Per compilare un programma, dapprima dobbiamo... scriverlo. Facciamo in modo che il programma risulti come nel riquadro **"Prova.ASM"**.

Il programma può essere scritto con un qualsiasi editor di testi. Dopo averlo scritto, registriamolo con il nome **PROVA.ASM** (l'estensione **.ASM** è di importanza fondamentale per la correttezza della procedura!). A questo punto sincerariamoci di avere a disposizione i tre programmi...

MASM.EXE
LINK.EXE
EXE2BIN.EXE

...e quindi, in caso positivo, lanciamo da Dos il comando...

MASM PROVA;

...che richiama Masm passandogli, come parametro, il nome del file PROVA (l'estensione .ASM viene presa di default). Il segno di punto e virgola (;) presente dopo il nome del file significa che tutti gli altri parametri (che, ripetiamo, sono numerosi) non sono importanti. L'assemblatore, pertanto, utilizzerà i parametri di default.

In caso di compilazione andata a buon fine, il computer visualizzerà un messaggio di conferma...

Microsoft (R) Macro Assembler
Version 5.10
Copyright (C) Microsoft
Corp 1981, 1988.All rights reserved.
50068 + 393321 Bytes symbol space free
0 Warning Errors
0 Severe Errors

...dopo aver creato il file **PROVA.OBJ**; con il comando...

LINK PROVA;

...il file oggetto PROVA.OBJ verrà finalmente convertito nel file eseguibile **PROVA.COM.**

Anche in questo caso va passato il nome del file senza estensione (il default è **.OBJ**, cioè "oggetto") ed il punto e virgola indica ancora che accettiamo le opzioni di default del LINK. In caso di successo, il PC visualizzerà:

Microsoft (R) Overlay Linker
Version 3.69
Copyright (C) Microsoft Corp
1983-1988. All rights reserved.
LINK: warning L4021:
no stack segment

Il **warning** (messaggio di avvertimento) prodotto (cioè: no stack segment) avverte che nel programma sorgente è stata notata l'assenza del segmento di stack, ma noi sappiamo che il programma è di tipo .COM, che non necessita della definizione di tale segmento (dato che viene preso uguale a quello del codice). Il messaggio di "pericolo", quindi, può essere tranquillamente ignorato.

Ciò non vale, ovviamente, nel caso si debbano produrre programmi di tipo .EXE, nei quali è obbligatorio dichiarare lo stack segment.

L'operazione di LINK produce un file eseguibile, però con estensione .EXE; deve ora deve essere trasformato in uno di tipo .COM, e questo può essere fatto tramite il comando:

EXE2BIN PROVA.EXE PROVA.COM

Prodotto il file .COM, si dovrà cancellare il file .EXE con un banale DEL PROVA.EXE; potremo quindi richiamare (ed attivare) il file appena compilato semplicemente di-

gitandone il nome da Dos (Prova + Return).

Se tutto è andato correttamente, il programma parte, e dovrebbe visualizzare sullo schermo un carattere "**a**" minuscolo. L'effetto è prodotto dalle due istruzioni:

```
mov AL,'a'
int 29h
```

La prima carica nel registro AL il carattere Ascii "a", mentre la seconda richiama una funzione del Dos che, appunto, scrive il carattere sul video, qualunque sia la scheda grafica utilizzata in quel momento.

Sbagliando si impara

Proviamo a generare volontariamente un errore, cambiando il registro AX nell'istruzione **mov ES, AX** in **AZ** in modo che l'istruzione diventi **mov ES, AZ**. Ricompiliamo il programma tramite l'ormai noto MASM PROVA;. Questa volta il compilatore segnalerà un errore:

Microsoft (R) Macro Assembler Version 5.10
Copyright (C) Microsoft Corp 1981, 1988.All rights reserved.
PROVA.ASM(11): error A2009: Symbol not defined: AZ
50068 + 393321 Bytes symbol space free
0 Warning Errors
1 Severe Errors

Il numero posto di fianco al nome del file (**11**) indica la riga del testo sorgente nella quale è stato riscontrato l'errore. Il file .OBJ non verrà prodotto e bisognerà ri- editare il testo sorgente per apportare la correzione.

Per semplificare la procedura di compilazione, si può creare un file batch, che richiami automaticamente i vari programmi necessari all'operazione. Il file sarà **ASM.BAT** e lo scriveremo come segue

```
MASM %1;
LINK %1;
EXE2BIN %1.EXE %1.COM
DEL %1.EXE
```

Per compilare un programma sarà sufficiente digitare...

ASM nomefile

...che richiamerà il file batch che attiverà tutti i programmi necessari alla compilazione del file sorgente.

Il Debug

Quando il testo sorgente diventa lungo e complicato, può non essere sufficiente osservarne gli effetti per cercare di correggere eventuali errori. Sarebbe utile, invece, eseguire il programma passo - passo, prender nota del valore delle variabili in gioco, dei registri, delle zone di memoria, inserire breakpoints (interruzioni volontarie), eseguire pezzi di programma, e tutto ciò che serve, insomma, per l'individuazione degli errori. Per questo viene in aiuto un potentissimo programma fornito assieme al Macro Assembler, dal nome **Code View Debugger** (CV), che consente di "debuggare" un programma proprio come se stessimo lavorando in Basic, ossia esaminando il testo SORGENTE(!), corredato di tutte le labels, le variabili, i commenti, eccetera.

Lavorando con **CV** si ha l'impressione di lavorare con un linguaggio ad alto livello, quali il TURBO-C o il QUICKBASIC: le possibilità offerte sono praticamente le stesse. Il programma **CV.EXE** ha bisogno di un file di supporto **CV.HLP**, che contiene gli help in linea in grado di aiutare l'operatore in caso di bisogno.

Per richiamare il debugger bisogna semplicemente specificarne il nome, seguito dal nome del programma da controllare.

Nel caso del programmino di esempio precedente, bisognerà digitare...

```
CV PROVA.COM
```

Immediatamente si presenterà la schermata del debugger, con il programma PROVA listato sullo schermo. Le varie istruzioni per l'uso del debugger saranno presentate, mese per mese, su queste pagine.

Operando nel modo descritto, però, il debugger visualizza il programma già nel formato eseguibile, ossia con le labels e le variabili sostituite dagli indirizzi assoluti; quindi, poco chiaro.

Per "vedere" il programma sorgente, dobbiamo compilarlo con alcune opzioni particolari, che sono **/Zi** per il MASM e **/Co** per il link. Le istruzioni per compilare il programma diventano pertanto:

```
MASM /ZI PROVA;
LINK /Co PROVA;
EXE2BIN PROVA.EXE PROVA.COM
DEL PROVA.EXE
```

A questo punto, con CV PROVA.COM si entrerà ancora nel debugger, ma in questo caso sarà il programma sorgente ad essere visualizzato, e non l'eseguibile.

La compilazione con le opzioni descritte aggiunge, nel file eseguibile (.EXE oppure .COM), le informazioni relative al listato sorgente necessarie per il debugger; il file risulta inevitabilmente (molto) più lungo.

Le opzioni sono da evitare non appena ci rendiamo conto che il programma funziona correttamente.

Alla fine delle operazioni il programma è da ricompilare secondo le istruzioni viste precedentemente.

LE ISTRUZIONI DEI PROCESSORI 80X86

La prima istruzione in esame si chiama **MOV**, e consente di leggere (e scrivere) nella memoria del PC e nei registri. Insieme a questa istruzione, tanto per restare sul classico, presenteremo anche quella di addizione, chiamata **ADD**, che serve per addizionare il contenuto di due locazioni di memoria, di due registri, di una locazione con un registro, e così via.
Nel caso di caricamento di dati da/verso un indirizzo in memoria, viene preso, come segmento di default, il registro **DS**. Se MOV deve prendere in considerazione un altro segmento, bisogna specificarlo **prima** dei parametri sorgente o destinazione, con il nome del registro di segmento desiderato seguito dal carattere di doppio punto (**:**), esempio:
MOV Seg1:Destinazione, Sorgente
oppure
MOV Destinazione, Seg2:Sorgente
Le parole **Seg1** e **Seg2** specificano il segmento che deve essere considerato nel caricare i dati da/verso l'indirizzo specificato; segmento che, di default, vale

Per "rompere il ghiaccio" con l'assembler è bene iniziare con l'effettuare semplici esperimenti con un'istruzione sicuramente molto usata non solo nel Linguaggio Macchina dei computer Ms-Dos compatibili, ma anche (nelle varie forme sintattiche che le competono) in qualsiasi altro linguaggio che "interagisce" direttamente con il microprocessore

MOV

MOV è il codice mnemonico (assembler) dell'istruzione, ed è l'abbreviazione della parola inglese **move**, ossia muovi.
Lo scopo di MOV è quello di muovere, o meglio, copiare, il contenuto di una zona di memoria in un'altra.

Destinazione. E' il registro o locazione di memoria nella quale andrà a finire il contenuto di ciò che viene letto da Mov. La destinazione può essere:
- Uno dei **registri** a 16 bit : **AX, BX, CX, DX** (anche nelle forme a 8 bit AH, AL, BH, BL, CH, CL, DH, DL), **SI, DI, SP, BP, CS, DS, SS, ES.**
- Una qualsiasi **locazione** di memoria (espressa come offset tra 0 e FFFF relativo ad un segmento)
- Un **puntatore** ad una locazione, espresso da un registro o da una coppia di locazioni.

MOV Destinazione Sorgente

Sorgente. E' il registro, la locazione di memoria, oppure il valore immediato che andrà a finire in Destinazione. Sorgente può essere:
- Uno dei **registri**: AX, BX, CX, DX (anche nelle forme a 8 bit AH, AL, BH, BL, CH, CL, DH, DL), SI, DI, SP, BP, CS, DS, SS, ES.
- Una qualsiasi **locazione** di memoria (espressa come offset tra 0 e FFFF relativo ad un segmento)
- Un **puntatore** ad una locazione, espresso da un registro o da una coppia di locazioni.
- Un **valore immediato**.

Attenzione a:

Nel caso di assegnazione tra registro e registro, **Destinazione** e **Sorgente** NON possono essere entrambi registri di segmento, quindi non sarà possibile un'istruzione del tipo MOV DS,CS. Bisognerà invece passare "attraverso" un altro registro di supporto.
L'assegnazione diretta tra un indirizzo di memoria ed un altro (esempio: **MOV Loc1,Loc2**) non è possibile.
L'istruzione MOV non influenza alcun FLAG.

Alcuni esempi di istruzione MOV:

MOV	**Buffer,**	**AX**	(Memoria, registro)
MOV	**CX,**	**Data1**	(Registro, memoria)
MOV	**DX,**	**SP**	(Registro, registro)
MOV	**BL,**	**10**	(Registro, val. immediato)
MOV	**Data2,**	**150h**	(Memoria, val. imm.)
MOV	**AL,**	**ES:Memory**	(Reg., mem. con assegnazione di segmen.)
MOV	**CL,**	**[BX]**	(Registro, memoria puntata da BX)

ADD

ADD è il codice mnemonico (assembler) dell'istruzione, ed in inglese significa "addiziona". Lo scopo di ADD è infatti quello di addizionare il contenuto di una zona di memoria ad un'altra.

Sorgente. E' il registro, la locazione di memoria, oppure il valore immediato che verrà addizionato a *Destinazione*. Anche *Sorgente* può essere uno dei tipi già esaminati per MOV.

ADD	Destinazione	Sorgente

Il segmento di default per l'istruzione ADD, nel caso di somma di dati contenuti in un indirizzo in memoria, è anche qui il registro **DS**. Per ciò che riguarda altri segmenti, valgono le stesse considerazioni fatte per MOV

Destinazione. E' il registro o la locazione di memoria nella quale finirà il risultato dell'addizione, ed è anche il primo operando dell'addizione stessa. *Destinazione* può essere uno degli elementi già esaminati in MOV.

Alcuni esempi di istruzione ADD:

ADD	**Buffer,**	**AX**	;Buffer = Buffer + AX (mem-reg)
ADD	**CX,**	**Data1**	;CX = CX + Data1 (reg-mem)
ADD	**DX,**	**SP**	;DX = DX + SP (reg-reg)
ADD	**BL,**	**10**	;BL = BL + 10 (reg-imm)
ADD	**Data2,**	**150h**	;Data2 = Data2 + 150h (mem-imm)
ADD	**ES:Memory,**	**AL**	;ES:Memory = ES:Mem. + AL (mem. ;con ass.seg.-registro)
ADD	**CL,**	**[BX]**	; CL = CL + (BX) (reg-mem.indiretta)

quanto contenuto in **DS**. Essi possono essere un valore numerico (compreso tra 0 e FFFF) oppure, meglio, un registro di segmento (CS, DS, SS, ES).

Istruzione Add

La seconda istruzione, che descriviamo brevemente, è anch'essa un'istruzione "classica" di un qualsiasi linguaggio Assembler.
Un esempio noto è il seguente:
ADD Seg1:Destinazione,Sorgente
oppure
ADD Destinazione, Seg2:Sorgente

Istruzione Adc

ADC è l'abbreviazione di **AD**d with **C**arry, ossia, addiziona con riporto. L'istruzione funziona esattamente come la precedente ADD, ed anche la sintassi, le varie forme, le avvertenze ed i flags influenzati, sono i medesimi.
L'unica differenza consiste nel fatto che ADC, oltre ad addizionare i due operandi *Destinazione* e *Sorgente*, considera anche il *Carry* (=Riporto), che è un bit contenuto nel registro dei Flags. Questo può essere utile nel caso si debba tener conto del risultato di una precedente operazione.

Attenzione a:

L'istruzione **ADD** esegue la somma **senza** tener conto del valore del riporto, o **Carry** (**CF** nel registro Flags).
Destinazione non può essere un registro di segmento (CS, SS, DS, ES).
La somma diretta tra un indirizzo di memoria ed un altro indirizzo (esempio: **ADD Loc1, Loc2**) non è possibile. Bisogna passare tramite un registro.
L'istruzione ADD, a seconda del risultato dell'addizione, influenza i seguenti FLAGS: AF, CF, OF, PF, SF, ZF.

```
;
; SOMMA.ASM :       Somma di due numeri
; Primo numero      =   locazione OP1
; Secondo numero    =   locazione OP2
; Risultato         =   locazione RESULT
;
cseg    SEGMENT PARA PUBLIC 'CODE'
        org 100h
        ASSUME cs:cseg, ds:dseg, ss:cseg, es:cseg

Start:
        mov     AX,CS
        mov     DS,AX           ;DS = CS
        mov     ES,AX           ;ES = CS

; Programma di somma
        mov     AL,OP1          ;Primo operando
        mov     AH,OP2          ;Secondo operando
        add     AL,AH           ;Addiziona AL con AH
        mov     RESULT,AL       ;Salva AL in RESULT

        mov     AH,4Ch
        mov     AL,0
        int     21h             ;Ritorno al DOS

; Dati utilizzati dal programma
        OP1     DB 10
        OP2     DB 15
        RESULT  DB 0

cseg    ENDS
        END     Start
```

Un semplice esempio

Come esempio non possiamo fare a meno di presentare un banale programma che si limita a **sommare due numeri**, anche perchè le istruzioni sinora descritte non ci consentono di andare oltre. Alla classica "intestazione" per i programmi COM (di cui ci siamo occupati in altro articolo), aggiungiamo altre istruzioni in modo che il programma sia identico a quello riportato nel riquadro *"Somma di due numeri"*, che ora descriviamo in dettaglio.
Tralasciando la parte di intestazione ed inizializzazioni varie (che supponiamo già note ai nostri lettori), resta da prendere in esame la parte relativa all'elaborazione vera e propria della somma e a quella di dichiarazione delle variabili.
La procedura di somma è semplicissima: dapprima viene caricato, nel registro AL, il contenuto della locazione di memoria OP1; poi si carica, in AH, il contenuto della locazione OP2; quindi si somma AH con AL. Il risultato (che resta in AL) viene trascritto nella locazione di memoria RESULT.
I dati OP1, OP2 e RESULT sono dichiarati tramite la **direttiva DB** (Define Byte, vedi le righe successive al commento *"Dati utilizzati dal programma"*), che riserva un'area di memoria lunga un byte e la assegna alla variabile eventualmente specificata prima della direttiva DB.
E' da notare che questa NON è un'istruzione assembler, ma semplicemente informa il compilatore di riservare, in quel punto del programma, un'area lunga un byte.
Il numero presente dopo DB rappresenta il valore di inizializzazione che sarà appunto trascritto nelle tre aree di memoria (ciascuna lunga un byte) all'atto della compilazione del programma, e che quindi sarà presente ogni volta che questo verrà richiamato da disco e fatto partire.
In pratica, i tre valori 10, 15 e 0 rappresentano i numeri che verranno interessati dall'operazione di addizione e che serviranno per verificare la correttezza dei nostri esperimenti.
Dopo aver digitato il programmino, registriamolo con il nome SOMMA.ASM e compiliamo il programma tramite le istruzioni:

```
MASM SOMMA;
LINK SOMMA;
EXE2BIN SOMMA.EXE SOMMA.COM
DEL SOMMA.EXE
```

...mandandolo in esecuzione digitando il semplice comando SOMMA.
A questo punto potremo vedere che... non succede un bel niente!
Il programma, ammettendo che la procedura di compilazione non abbia fornito errori, funziona sicuramente, solo che gira per "i fatti suoi": cioè somma il contenuto di OP1 a quello di OP2 e, subito dopo, pone mette il risultato della somma in RESULT senza mostrare alcunchè sul video (non vi sono, infatti, istruzioni di visualizzazione); ecco perchè non ci accorgiamo dell'effettivo funzionamento del programma.
Dato che, da DOS, non è possibile controllare direttamente il contenuto di locazioni di memoria, dobbiamo usare un secondo programma di aiuto, ossia il Code View Debugger (CV.EXE).

Debug del programma

Prima di richiamare il debugger, dobbiamo ricompilare il programma usando le opzioni del MASM e del LINK che forniscono, al debugger, le informazioni necessarie per visualizzare il listato sorgente, comprese le variabili ed i commenti relativi.

```
MASM /Zi SOMMA;
LINK /Co SOMMA;
```

Questa volta non trasformiamo il file .EXE in .COM perchè, altrimenti, perderemmo le informazioni sul sorgente da passare al debugger. Richiamiamo direttamente **CV** con il comando CV SOMMA.EXE.
Il programma presenterà una schermata visualizzando in verticale, sulla destra, tutti i **registri** del processore ed i relativi contenuti in esadecimale; in basso compare una finestra per i **comandi immediati,** mentre il resto dello schermo sarà

occupato dal **listato sorgente**. Tramite il tasto **F6** sarà possibile spostarsi dalla finestra immediata a quella del sorgente. In quest'ultima, con i soliti tasti cursore (PgUp, PgDn, ecc.) potremo spostarci a piacere lungo il listato sorgente, mentre nella prima potremo impartire comandi a CV.

Facciamo ora partire il programma SOMMA.

Per fare ciò premiamo i tasti **ALT + R**, che produrrà la visualizzazione di un menu a tendina; quindi scegliamo l'opzione **Restart** per posizionare il registro **IP** (=Instruction pointer) all'inizio del programma (che ricordiamo, è posto al valore 0100h dalla direttiva ORG), poi di nuovo **ALT + R** e finalmente l'opzione **Start**, che farà partire effettivamente il programma. Dopo un istante, se tutto va bene, verrà visualizzato un messaggio che informa che il programma è terminato, e potremo controllare il lavoro svolto esaminando direttamente le locazioni di memoria ed i registri, dal momento che il programma agisce su questi.

Se l'operazione di somma è stata svolta correttamente (e DEVE essere così, a meno di errori di digitazione nel listato sorgente), nella locazione RESULT sarà presente il valore esa **25**, somma di OP1 (esa **10**) e OP2 (esa **15**).

Per fare ciò dobbiamo utilizzare uno dei molti comandi presenti nel debugger, ad esempio: **D start end**, che produce la visualizzazione del contenuto di una parte di memoria, dall'indirizzo **start** all'indirizzo **end**.

La visualizzazione è sempre in esadecimale.

Dato che stiamo lavorando in simbolico, ossia con il listato sorgente, *start* e *end* potranno essere espressi, invece che da valori di indirizzi, dai nomi delle variabili definite nel programma.

Spostiamoci nella finestra immediata (tasto **F6**) e digitiamo **D OP1 OP1**.

Con questo comando diretto chiediamo a CV di mostrare il contenuto della memoria dalla locazione OP1 alla locazione OP1 (in altre parole, solo OP1).

Dovremmo ottenere il numero **10**, valore iniziale impostato all'inizio dal nostro programma.

Con **D OP2 OP2**, come intuitivo, otterremo **15**, mentre con **D RESULT RESULT** si visualizzerà il risultato, ossia **25**. Il debugger, a fianco del valore letto dalla locazione, visualizzerà (sempre in esadecimale) anche il valore del **segmento** e del relativo **offset** della locazione interessata.

Il registro **CV** prenderà, come segmento di default per i dati, il registro **DS**.

Se, per qualsiasi ragione, è necessario leggere i dati da un altro segmento, lo si potrà specificare prima degli indirizzi *Start* o *End*, seguito da un carattere doppio punto (:) come ad esempio...

D Seg:start Seg:end

Seg può essere un valore direttamente espresso in decimale oppure uno dei registri di segmento CS, DS, SS, ES.

Proviamo ora a cambiare i valori delle locazioni OP1 e OP2, per modificare il risultato della somma.

A questo scopo serve un'altra istruzione del registro CV, cioè **E** (Enter) per cambiare il valore contenuto nelle locazioni di memoria. Il formato è il seguente:

E addr.

...in cui **addr** è la locazione della quale vogliamo cambiare il valore.

Il comando **E** produrrà la visualizzazione del segmento e dell'indirizzo relativo alla locazione specificata, con a fianco il vecchio valore; permetterà di cambiarlo semplicemente introducendo, da tastiera, il nuovo e premendo il tasto Enter.

Con il comando...

E OP1

...inseriamo, ad esempio, il valore **12**. Con... .

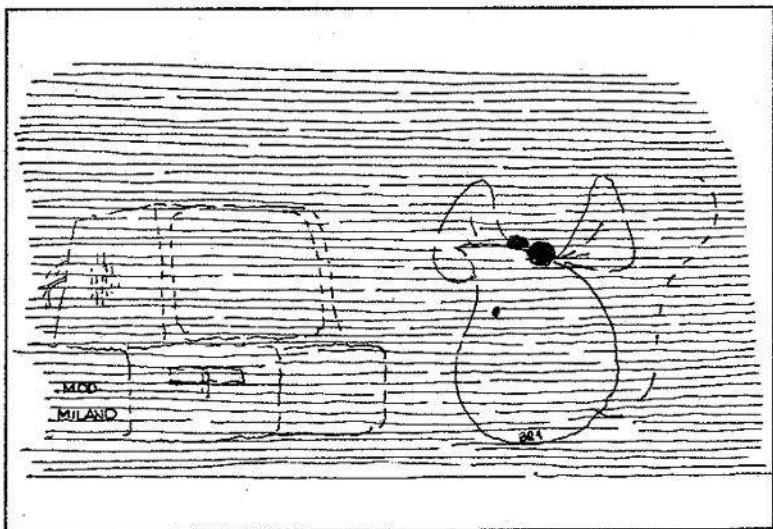
E OP2

...inseriamo invece il valore **23**. Premiamo **ALT - R** e scegliamo l'opzione **Restart** per riposizionare IP all'inizio del programma (si può vedere IP visualizzato a destra, nell'area dei registri, che in questo caso vale 0100h).

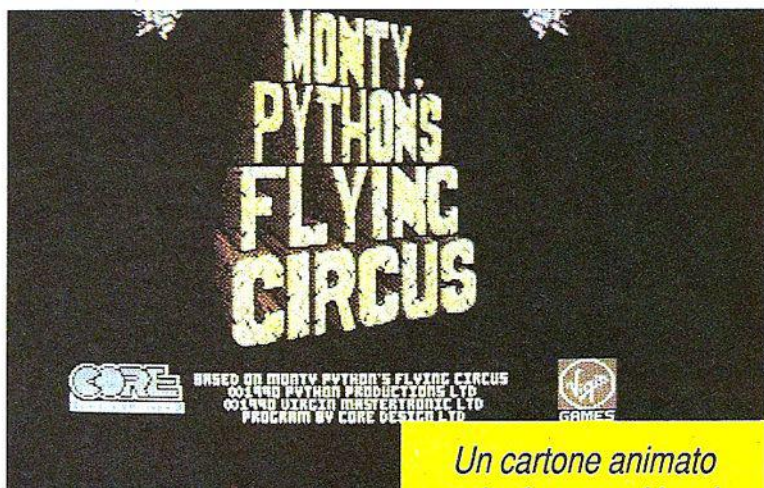
Questa volta, però, non facciamo partire il programma con ALT-R e Start, perchè l'azione ricaricherebbe il programma da disco, e quindi **OP1** e **OP2** verrebbero nuovamente inizializzati a 10 e 15.

Eseguiamo, invece, il programma passo-passo sino alla fine agendo sul tasto **F10**. Ogni volta che lo premiamo, IP avanza di un'istruzione riga sul listato sorgente, e potremo controllare l'esecuzione osservando il contenuto dei registri che cambiano di valore ad ogni pressione di F10. Alla fine dell'esecuzione (messaggio di programma terminato normalmente) possiamo ancora una volta controllare il risultato tramite il comando **D RESULT RESULT**, che questa volta visualizzerà **35**.

Il comando **HELP**, impartito in qualsiasi momento, visualizzerà l'help del debugger, molto esteso e comprendente spiegazioni dettagliate (pur se in inglese) per tutti i comandi e le opzioni presenti in CV.

Infine il comando **QUIT** consentirà il ritorno in ambiente DOS.

MONTY PHYTONS FLYING CIRCUS

Un cartone animato molto famoso si lancia verso mille avventure, tutte "realistiche"

Computer: Amiga inespanso
Tipo: Arcade multifasse
Controllo: Joystick
Softhouse: Virgin/Mastertronic

Il protagonista è un personaggio ben noto agli amanti dei fumetti anglosassoni firmati da Terry Gilliam, mentre il programma è un piccolo capolavoro di programmazione destinato a segnare veri standard di produzione nel suo campo.

Il gioco

Gumby, il protagonista, deve vedersela con un'avventura multischermo ed un numero incalcolabile di puzzle e rompicapo da risolvere, solo qualche volta con l'ausilio della pura velocità di manovra del joystick.

Non si tratta di un normale arcade game stile "a piattaforme" alla **Manic Miner**, ma di una serie incredibile di scene di gioco, la cui originalità è senza dubbi notevole.

In particolare, il gioco è ravvivato dalla possibilità di Gumby di trasformarsi da omone saltellante in pesce, per attraversare fasi a "movimento completo" sullo schermo.

Il gioco è, materialmente, piuttosto semplice, e da qui arriva una parte del suo fascino. Inoltre i puzzles da risolvere sono veramente coinvolgenti, oltre che divertenti in senso stretto per il sottile humor che spesso pervade la situazione. Il tutto incentiva a proseguire per potere "vedere lo schermo dopo", come in ogni gioco del genere ben riuscito.

La tecnica

La grafica è molto simile a quella dei cartoni animati originali. L'uso dei colori e delle ombreggiature è di qualità notevole e ricalca bene gli episodi televisivi. Gli effetti sonori, campionati talvolta direttamente dalla serie TV, sono eccellenti e contibuiscono a render l'atmosfera di gioco ancora più realistica, per quanto si possa usare questo termine per un cartone animato. Si va dalle melodie di **John Cleese**, sino alle campionature di voce del doppiatore inglese Terry Jones. Il controllo via joystick è perfettamente all'altezza del compito.

Il voto

Un ottimo gioco a piattaforme, coinvolgente, tecnicamente ottimo e ricco di incentivi. 9

POWER BOAT

Una gara esaltante tra motoscafi, sponsorizzata dal famoso Don Johnson

Computer: Amiga inespanso
Gestione: Joystick
Tipo: Simulazione
Softhouse: Accolade

Il gioco

Sei difficili percorsi tra Miami Beach e San Francisco fanno da sfondo a questa appassionante simulazione di gara offshore.

Ogni percorso è indicato da una coppia di boe attorno alle quali i gareggianti devono veleggiare(?) con i propri fuoribordo. Si può usare la bussola per la navigazione, riportata sulla accurata strumentazione di bordo, oppure una mappa (che richiama il ben noto "Flight Simulator") sullo schermo.

La visione del gioco è normalmente in prospettiva con vettori pieni, vista ovviamente dagli occhi del pilota. Vi sono numerosissime opzioni che consentono di personalizzare il gioco e di aumentarne l'altrimenti monotona routine.

La tecnica

La grafica, in soggettiva, avrebbe potuto essere resa più veloce e fluida diminuendo il numero di poligoni tracciati per rendere la barca del giocatore o gli oggetti, caratteristica interessante tecnicamente, ma che comporta una perdita di look decisamente notevole.

La strumentazione di bordo è molto ben disegnata ed occupa buona parte dello schermo.

La porzione di video animato può così essere gestita più rapidamente da Amiga, essendo piuttosto ristretto rispetto agli standard.

Gli effetti sonori sono limitati quanto quelli effettivamente percepibili da un pilota che guida un offshore.

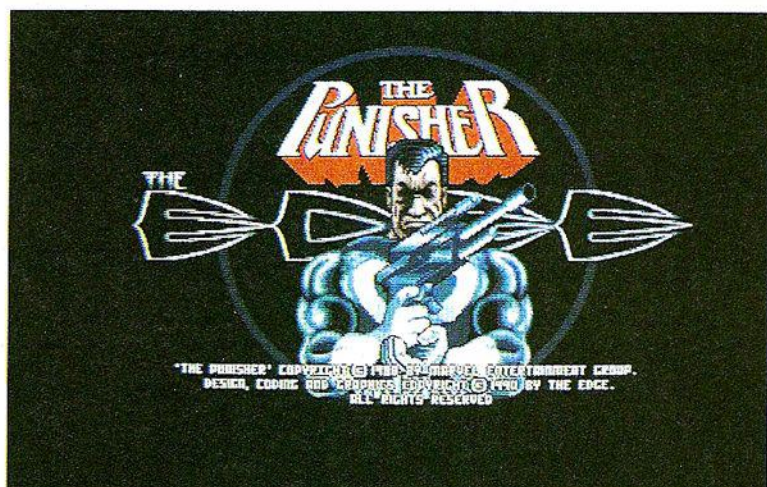
L'interfaccia utente è semplice e curata.

Il voto

Una buona simulazione, per gli amanti del genere. 8.

THE PUNISHER

Come usare un criminale per eliminarne altri; polizia permettendo

Computer: Amiga inespanso
Gestione: Joystick
Tipo: Arcade di ruolo
Softhouse: The Edge

Il Punitore è un personaggio dei fumetti Marvel ben noto anche in Italia, il classico eroe ammazzatutti per vendetta. Il videogioco è un arcade comprendente azioni e fasi manageriali

❑ ❒

Il gioco

Frank Castle, dopo avere ucciso ben duecento criminali, è stato messo in galera, ma il direttore del penitenziario pensa di sfruttarne le capacità inserendolo in un gruppo segreto chiamato **"Thrust"**, teoricamente nato per fare giustizia sommaria dei criminali. Frank si accorge però presto che anche il Thrust è una banda di criminali camuffata, quindi decide di eliminarli e da qui parte il nostro videogioco.

L'ambientazione è una serie di schermate statiche che raffigurano vari ambienti: l'interno del furgone supercomputerizzato del Punitore, le fogne citadine, ville, bassifondi eccetera. Ad ogni schermata bisogna eliminare i criminali che compaiono tramite i colpi sparati dalle armi che il Punitore porta sempre con sè. La prospettiva del gioco è come quella di **Operation Wolf**.

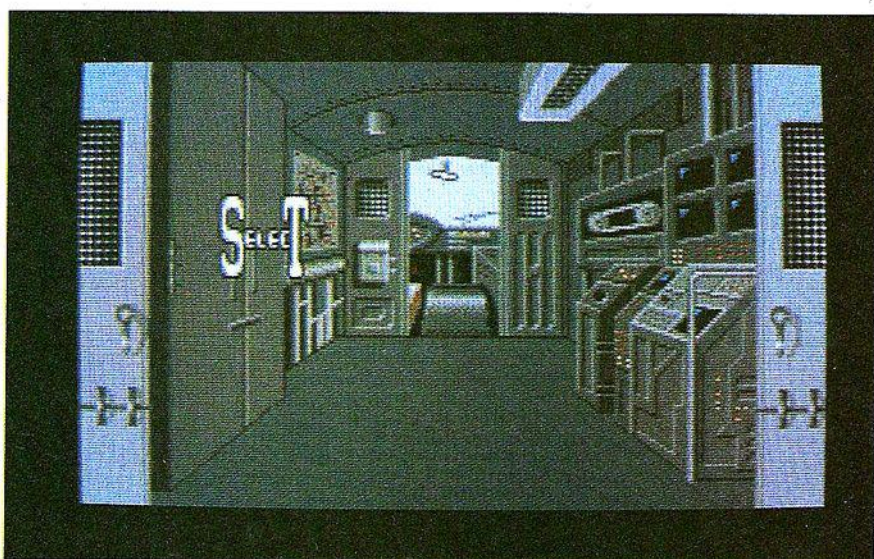
L'interesse principale del programma è, comunque, non solo nelle fasi di combattimento vero e proprio, bensì nelle fasi di scelta della strategia di battaglia e delle armi da utilizzare.

La tecnica

La grafica è stata realizzata avvalendosi dei disegnatori originali Marvel del Punitore, quindi estremamente simile a quella dei fumetti veri e propri.

L'animazione non è effettivamente delle migliori, ma la grafica di scena è veramente di ottima fattura. Gli effetti sonori sono pochi, ma ben realizzati; l'interazione con il giocatore è semplificata dall'ambiente ad icone.

Il voto

Un gioco decisamente interessante, non solo tuttogrilletto. 8.

di Domenico Pavone

DUE PERIFERICHE UN PO' SPECIALI

Stavolta accontentiamo tanto gli utenti di A-500 che quelli del più evoluto A-2000, esaminando due periferiche molto interessanti

In queste pagine parliamo di un drive esterno utilizzabile con qualsiasi modello Amiga (ma particolarmente adatto ai 500), e di un modem. E fin qui niente di nuovo, direte.

Errore. Qualcosa di speciale c'è, e proprio a livello macroscopico. Perchè, signori, ci si sta riferendo a un **drive per floppy disk da 5.1/4**, e a un modem INTERNO per Amiga 2000!

Drive Cumana Cax1000s

L'uso di un drive da 5.1/4, su Amiga, è normalmente confinato alla compatibilità con il mondo Ms-Dos, quale che sia il metodo per raggiungerla: schede hardware, o emulatori software.

In tal senso, il drive Cumana non fa eccezione, consentendo l'accesso hardware a quello standard di formattazione dei floppy, ma ciò che lo contraddistingue in positivo, è la **completa adattabilità alla trackdisk device** di Amiga.

In altre parole, il Cumana può, sì, funzionare come un drive per Pc, ma anche come se fosse un qualunque drive esterno di Amiga!

Esternamente si presenta molto compatto e lineare, dotato di un cavetto che va inserito nella porta riservata ai drive esterni.

Sul retro, nota importante, è presente un connettore per il collegamento di altri drive, per cui può essere adoperato come primo supplementare (**df1:**) negli Amiga 500, con la possibilità di allacciare al connettore un altro drive che diventerà **df2:**.

Ancòra sul retro c'è una chance utile soprattutto a chi dispone di soli 512 KB di ram: un bell'interruttore a levetta in grado di escludere la periferica.

Com'è noto, infatti, ogni drive presente implica l'impegno di una certa quantità di memoria. Cosa che, in condizioni critiche può significare la necessità di scollegare fisicamente un eventuale drive esterno.

Sul frontalino, oltre alla feritoia per l'inserimento dei dischi e l'immancabile spia luminosa, troviamo un altro selettore che riporta la scelta 40/80. Il suo significato è immediatamente riferibile al formato dei floppy cui accedere: **40** oppure **80 tracce**.

La selezione sui due formati, inoltre, fa da switch vero e proprio tra gli standard Amiga e Pc. Se si predispone il drive su 80 tracce, si inserisce un floppy da 5.25 nel drive (la levetta di chiusura può essere abbassata solo se è presente un disco al suo interno) e quindi si lancia Amiga con il solito disco Workbench, sullo schermo apparirà l'icona di quel disco, eventualmente caratterizzata dalla dicitura "Bad" se non è ancora formattato, o comunque formattato non in modo Amiga. Da Shell (o Cli), un analogo comando Info esibirà tutte le informazioni che riguardano la nuova unità.

Da questo momento, ogni operazione sul disco è assolutamente **identica** a quanto applicabile ai comuni drive da 3.5, tanto da Workbench che da ambiente Dos: formattazione, copia, apertura di icone, directory e subdirectory, e senza alcun problema di velocità!

Il floppy da 5.25 sarà insomma identico a qualunque altro da 3.5, reagendo ottimamente anche ad utility come Disk Salve o DiskDoctor, ad ottimizzatori come il Disk Arranger, nonchè a qualunque disk editor per Amiga.

Nella compatibilità è anche incluso il riconoscimento automatico del drive da parte del sistema, senza la necessità di "mountare" la periferica, nonchè il riconoscimento del cambio di disco: se si espel-

le il floppy dal drive e se ne inserisce un altro, la nuova icona apparirà sullo schermo, come avverrebbe con un qualunque altro drive di Amiga.

Unica esigenza nell'uso di questa modalità (del resto piuttosto ovvia), quella di utilizzare floppy rigorosamente in doppia densità e doppia facciata, evitando i "supereconomici" senza alcuna etichetta, peraltro tranquillamente utilizzabili nel modo 40 tracce.

Tutto ciò porta a considerare il Cumana anzitutto come un drive di Amiga, nonostante l'inconsueto formato dei floppy: la Commodore, infatti, non offre un drive da 5.25 per il piccolo Amiga, ma solo modelli da 3.5 pollici.

Tuttavia, trova la sua principale ragione di essere quando si ha la necessità di **rapportare l'Amiga (soprattutto l'A-500) all'ambiente Ms- Dos**. In questo caso si hanno due possibilità.

Per trasferimento di files, o anche formattazione in rapporto all'uso di mini-emulatori come il **Dos2Dos** per Amiga, può essere mantenuto il formato ad **80 tracce**, corrispondenti a **720 KB**.

In questo caso, dovendo poi utilizzare il disco sui drive di un Pc, sarà necessario ricorrere a macchine dotate di **driver** particolari per leggere l'inconsueto formato (quantomeno su dischi da 5.25).

Per risolvere il problema alla radice, viene fornito in dotazione un floppy Ms-Dos contenente, per l'appunto, un device-driver di nome **Disk.sys** per consentire l'eventuale adattamento.

Utilizzando il drive, invece, su **40 tracce**, si avrà un drive pronto per l'uso sotto Ms-Dos. Da Workbench (o tramite Info della Shell) il drive non verrà più riconosciuto come drive Amiga, ma potrà essere adoperato da qualunque **emulatore hardware o software di ms-dos**, garantendo la compatibilità con il più diffuso standard di formattazione di quell'ambiente, vale a dire **40 tracce per 360 KB**.

La cosa può rivelarsi molto utile, come intuitivo, per chi ha necessità di adoperare computer Ibm compatibili **non** dotati di drive da 3.5.

Sebbene l'uso di questo drive sia indipendente dal modello Amiga che si possiede, nella pratica risulta più vantaggioso sui modelli A-500, potendo diventare ipso fatto il **drive b:** tanto per programmi come il **Transformer**, quanto per le mini-schede hardware **Xt** ed **At**, presenti sul mercato da tempo ma, anche queste, non fornite dalla Commodore (per ciò che riguarda l'A-500). Sui 2000, invece, resta intatta la sua capacità di utilizzo come drive Amiga, o per il trasferimento di dati. C'è da dire che chi possiede schede Ms-Dos (come le **Janus**) dispone già di un drive da 5.25: l'acquisto del drive di cui parliamo non rappresenta un vantaggio reale dal momento che il problema di disporre dei due formati (3.5 e 5.25) è già risolto dalle schede originali Commodore.

Inutile aggiungere che, anche senza ricorrere ad emulatori di qualsiasi genere, la gestione del drive in modo 40 tracce può essere anche affidata alle utility presenti nella directory **PCUtil** del disco **Extras**.

Supramodem 2400ZI

Nel settore hardware, la Supra Corporation è sicuramente una delle più note produttrici di accessori per Personal Computer.

In rapporto a questi ultimi, la Supra offre una gamma di soluzioni rivolte alla fascia ormai divenuta uno standard di accesso, quello dei **2400 baud**, in grado di soddisfare le esigenze di qualunque "amigo" telematico... non in vena di cavare troppi soldi di tasca. In particolare, oltre al classico e più diffuso modem esterno da allacciare via cavo seriale alla porta **RS-232**, la stessa casa commercializza due modelli **interni**, uno per i **Pc** ed uno per **Amiga**, il **2400zi**, al quale daremo un'occhiata più da vicino.

Un modem "interno", in pratica, non è altro che una scheda da infilare in uno degli slot di Amiga 2000, unico modello al quale è (ovviamente) riservato. Oltre alla scheda, la superimbottita confezione del Supra 2400zi comprende anche una piccola staffa matallica alla quale fa corpo un circuitino stampato dotato di tre prese per spinotti telefonici a "clip", un cavetto per la connessione alla linea telefonica, nonchè un altro cavetto corto già inserito in uno dei clip. All'hardware, si accompagnano un **floppy** non bootabile (ma sfruttabile interamente da Workbench, come vedremo tra breve), nonchè due esaurientissimi **manuali**: uno riservato all'installazione fisica della scheda, ed uno più generale (e più corposo: circa 80 pagine) dedicato alla gestione telematica del modem. Per finire, vi si trovano anche alcuni opuscoli divulgativi delle principali reti telematiche americane più o meno amatoriali (Compuserve, Genie, ecc.). Ma veniamo al sodo.

La scheda modem, fornita anche di un suo piccolo **altoparlante** per un eventuale ascolto diretto, si installa con estrema facilità e rapidità, aiutati in questo dalle chiare illustrazioni esibite dal manualetto dedicato. Tutto ciò che occorre fare è aprire il cabinet di Amiga 2000, ed infilare in uno degli slot disponibili il modem, con i componenti rivolti verso destra. Impossibile comunque sbagliare, in quanto è accessibile una sola posizione di inserimento. Dopodichè è necessario sfilare una delle staffe metalliche alloggiate sul pannello posteriore del computer, e sostituirla con la staffa dotata di connettori a clip fornita con la scheda.

Il tutto si traduce in pratica nello svitare una sola vite e nel riavvitarla dopo la sostituzione. C'è da dire che, nel caso l'Amiga sia già "satollo" di hard disk, espansioni, o altro, è possibile decidere se montare la staffa in una delle sette "feritoie" tradizionali, oppure nelle due poste a ridosso dell'alimentatore, più vicine ai drive, per intenderci.

Effettuata l'operazione non resta che aggangiare (con una semplice pressione) al modem il corto cavetto di connessione, che fuoriesce dal circuitino solida-

le alla staffa, ed il gioco è fatto. Richiede più tempo l'apertura del cabinet, che non la vera e propria installazione del modem.

Richiuso il computer, si disporrà ora sul posteriore dei due connettori a clip nei quali inserire (indifferentemente) il cavetto fornito in dotazione per agganciarsi alla linea telefonica, ed eventualmente anche quello proveniente da un apparecchio telefonico, che risulterà agibile in qualunque momento (anche a computer spento).

A questo punto, tutto è pronto per disporre di un modem di ottime prestazioni, integralmente compatibile con l'ormai indispensabile **Standard Hayes**. Tutto pronto... o quasi.

Il Supra 2400zi, infatti, necessita di qualche preliminare manovra software, peraltro molto semplice da effettuarsi anche se non si è molto esperti.

Il modem ha infatti bisogno di adoperare un suo device, di nome **Modem0.device**, che dovrà trovarsi nella directory **Devs:** di sistema. In altre parole, è da installare nella directory Devs del disco comunemente adoperato per lanciare il programma di comunicazione preferito.

Chi volesse può anche provvedere da sè prelevando il Modem0.device dal floppy in dotazione al modem, ma la Supra ha ben pensato di rendere tutto più facile: basta lanciare il sistema con il disco di boot che si adopererà in occasione delle proprie attività telematiche (può anche essere una copia del Workbench), accedere al Workbench, aprire l'icona disco del floppy Supramodem e biclickare su **InstallDriver:** penserà a tutto questo programma.

Nello stesso floppy, sono presenti inoltre due utility, entrambe di una certa importanza. A seconda del programma di comunicazione che si adotterà, può infatti sorgere un problema

Ovvio che, per connettersi via modem alla linea telefonica, si adopererà un programma terminale di propria scelta, e per Amiga ve ne sono molti. I più vecchi o meno evoluti tra questi, cercheranno senza alternative di adoperare la normale **Serial.device**, piuttosto che la nuova Modem0.device.

Programmi come il **JrComm**, invece, prevedono la scelta (nel caso specifico, **Generals** dal menu **Options**) del device: basterà settare con "modem0.device" l'opzione inerente il "driver" da adottare per la gestione della porta seriale, e tuto si concluderà lì.

Se invece il programma non prevede una simile opzione (p. es. **Diga**, **Ncomm**, ecc.), bisognerà sostituire i richiami alla serial device in esso presenti con analoghi richiami alla modem0.device.

Più facile a farsi che a dirsi, basta da Workbench clickare una volta sull'icona **ModemModify** quindi, tenendo abbassato il tasto shift, biclickare sull'icona del programma di comunicazione che si intende adottare (su una copia, non si sa mai!).

L'eventuale operazione va effettuata come ovvio una sola volta, il programma potrà poi essere adoperato senza altre difficoltà.

La stessa procedura, per inciso, può anche essere adoperata da Cli o Shell (nel caso il vostro programma-terminale non fosse dotato di icona), seguendo le dettagliatissime indicazioni fornite dal manuale di installazione.

Le "manovre" indispensabili sono tutte qua, ma nel disco c'è ancora un'icona: **Showmodem**. Stavolta si tratta di un'applicazione non obbligatoria, ma decisamente comoda.

Normalmente, infatti, i modem esterni sono dotati di una serie di **spie luminose** che forniscono le principali indicazioni sullo stato del modem e sull'andamento della trasmissione dati.

Come ovvio, tali indicazioni non possono essere implementate via hardware in un modem ben chiuso all'interno del nostro Amiga 2000 (a meno di non prevedere sforacchiamenti vari...).

Il programma in questione, in pratica, ne farà le veci. Conviene trasferirlo (anche tramite la sua icona da ambiente Workbench) nel disco contenente il programma di comunicazione, ed attivarlo (o farlo attivare dalla **Startup Sequence**) assieme a quest'ultimo. Si avrà così disponibile una finestrella con le consuete "spie" luminose che si illumineranno indicando gli stati **HS** (Alta velocità=2400 baud), **CD** (Carrier Detect), **RD** e **SD** (ricezione o trasmissione dati), **TR** (Terminal ready) e **MR** (Modem ready).

Per avere questa finestra sempre visibile sullo schermo del programma terminale (e non su quello restrostante, come avverrebbe normalmente), è poi possibile inserire nella Info della sua icona (o nel relativo comando Shell) il nome dello schermo al quale collegarla.

Testato a fondo, il modem si è rivelato assolutamente affidabile ed anche "veloce", pur nell'ambito dei 2400 baud, probabilmente grazie proprio al suo personale device di gestione della porta seriale.

E' dotato, poi, di una sua memoria interna non volatile, nella quale è possibile memorizzare una configurazione di default attiva allo start. Opzione, questa, comunque non indispensabile, in quanto, di solito, affidata alla fase di inizializzazione dei programmi di comunicazione.

Il set di comandi Hayes è completo, e non staremo qui a dilungarci su un tema ampiamente discusso nelle pagine riservate a questo argomento.

Unico positivo dettaglio da aggiungere, la sua ottima intercettazione del segnale Busy (occupato), purchè emesso dalle nuove centraline elettroniche Sip: segno, questo, di una qualità hardware di tutto rispetto...

In definitiva, il SupraModem 2400zi si dimostra pienamente all'altezza della situazione, senza alcun complesso di inferiorità per i modelli esterni, e molto comodo soprattutto per chi, grazie al non certo "mingherlino" Amiga 2000, dispone di poco spazio sulla scrivania di lavoro.

di Domenico Pavone

C1 - TEXT V.3, UN W/P TUTTO ITALIANO

C1-text ricomincia da 3. Sempre più sulla breccia il potente Word Processor per Amiga made in Italy

A distanza di più di un anno (CCC n. 70), rieccoci a parlare di C1-Text rivolgendoci non solo ai vecchi estimatori del prodotto, ma anche ai nuovi adepti del mondo **Amiga**, ansiosi di "guardarsi intorno" alla ricerca del software più congeniale alle proprie esigenze.

Escludendo la più diffusa tra queste, fatta di joystick e mouse a tutta birra (si sta parlando di games, si era capito?), senza troppi preamboli C1-Text assolve più che degnamente ad una delle prime che si presentano all'utente normale: la stesura di testi. Che, ovviamente, deve risultare comoda che più comoda non si può: sarebbe pura follia pensare di adoperare Amiga come sostituto di una semplice macchina da scrivere!

Sotto questo aspetto, e sin dalla sua prima release, il C1 ha subito ben definito le sue potenzialità, concentrandosi soprattutto su un obiettivo: quello di adoperare senza troppi sforzi la lingua italiana, e rendere il più semplice e completa possibile la manipolazione dei testi, o di loro singole sezioni. Questo, dichiaratamente, a scapito di prestazioni grafiche spesso collegate ad altro software di word processing.

C1-TEXT V.3.0

Prodotto da:
CLOANTO ITALIA
Via G.B.Bison 24
33100 UDINE

La versione **3.0**, come le sue precedenti, si limita infatti a consentire il caricamento e relativa visualizzazione di una immagine grafica in formato **IFF** (compresso o meno che sia), nonchè una sua eventuale stampa su carta

L'immagine, però, resta confinata in una sua finestra esclusiva, senza la possibilità di adoperare contemporaneamente grafica e testo in una stesso ambiente.

Utile, insom ma, quasi come una utility a sè stante, da adoperare eventualmente per inserire testate grafiche su carta, o a modificare i colori di una schermata grafica mentre si sta lavorando ad un testo, ma nulla di più. Dove invece il C1 Text la fa da padrone, è nel trattamento dei testi.

Ambienti paralleli

Intanto, come promesso nella versione **1.0**, e come di fatto già implementato dalla **2.0**, il programma consente un editing in **multitasking**. Il che comporta la possibilità di sfruttare più finestre video, ognuna autonoma e liberamente ridimensionabile, nelle quali eventualmente caricare documenti diversi o anche scambiare sezioni di testo tra una finestra e l'altra.

Per chi non fosse già avvezzo all'uso di un **Text Editor**, chiariamo meglio il concetto: supponiamo di avere scritto (o caricato da disco) un testo qualunque nella finestra principale di C1-Txt, e di volerne trasferire una sua parte in un

altro documento, lettera, o similare. Niente di più semplice: basta selezionare dal menu **Generale** la voce **Nuova finestra testo**, e verrà aperto un secondo ambiente di lavoro, di dimensioni inferiori, ma estendibile come qualunque altra finestra di **Intuition** attraverso l'angolo inferiore destro, evidenziato da un diverso colore (giallo, di default).

Per inciso la finestra principale, oltre ai comuni gadget di profondità che consentono (per esempio) di portare in secondo piano lo schermo mostrando quello del Workbench, dispone di altri due gadget speciali, riservati proprio ad eventuali altre finestre del C1 aperte in contemporanea. Il primo porta, in primo piano, la finestra principale, l'altro riesibisce tutte le eventuali finestre secondarie. Molto comodo, ma... rientriamo dalla digressione.

Dalla finestra principale, è possibile delimitare un qualunque blocco di testo attraverso l'omonima opzione dal menu **testo**, o anche eliminarlo fisicamente tramite l'opzione **Scorporare Blocco**. In entrambi i casi l'operazione va svolta adoperando il mouse, semplicemente facendolo scorrere sulla parte interessata e mantenendo abbassato il pulsante sinistro.

Quanto delimitato, o scorporato, viene memorizzato in un buffer, così se poi ci si posiziona nella seconda finestra, vi si clicka dentro per attivarla, e si seleziona **Inserire blocco** dallo stesso menu, ecco che il testo prima selezionato apparirà nella posizione voluta. Inutile aggiungere che si possono aprire più finestre, ognuna gestibile in maniera del tutto autonoma.

Muoversi in fretta

Visualizzando il menu **testo**, ci si è già imbattuti in una ricchissima dotazione di comandi dedicati alla manipolazione di blocchi di testo, ma anche alle indispensabili funzioni di **ricerca**, modifica dello **stile**, ed altre cui accenneremo tra breve. Da sottolineare la particolare "filosofia" adottata dal C1-Text per gestire l'interfacciamento con l'utente. E' possibile selezionare le opzioni dal tradizionale menu sulla barra-schermo, ma quasi tutte sono anche accessibili da tastiera, secondo associazioni mostrate nei menu (oltre che sul meticolosissimo manuale).

In particolare, le più frequenti possono essere adoperate ricorrendo ai tasti funzione, con una peculiare caratteristica di C1: se, per esempio, si vuole cancellare un blocco di testo, è necessario premere DUE volte il tasto **F2**, quindi evidenziare la sezione da eliminare con il mouse. Analogamente, i vari **spostamenti** ad inizio / fine documento, la **marcatura** di un punto del testo, ma anche la stampa su carta, seguono lo stesso tipo di procedura: una doppia pressione dei tasti funzione; evitando così, tra l'altro, involontarie attivazioni. Per chi fosse un patito del mouse, le stesse scelte possono essere effettuate direttamente sul video, che mostra, nella parte bassa, 10 icone, ognuna corrispondente ad un tasto funzione, attivabili clickandovi sopra con il mouse.

Le icone, notevolmente migliorate nella loro estetica rispetto alla prima versione del text editor (dalla 1.0, per intenderci), possono anche essere eliminate dallo schermo ricorrendo in qualunque momento al menu **Parametri / interfaccia utente**, che controlla numerose "feature" tutte molto utili pur se facoltative.

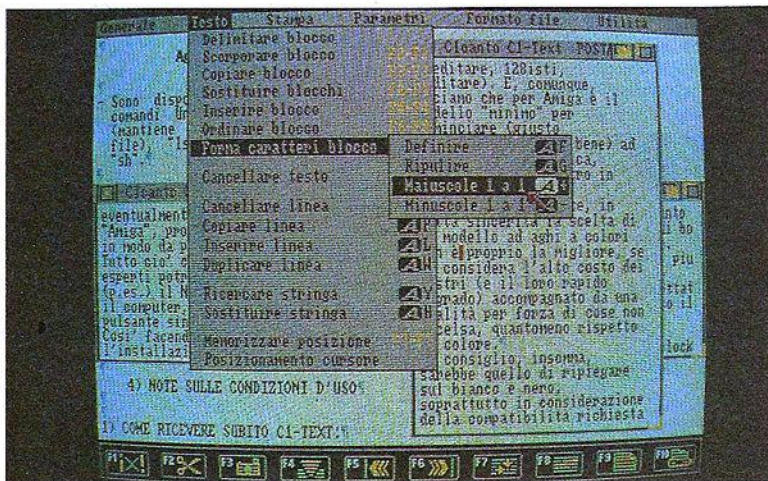
I normali **spostamenti** lungo il testo sono poi affidati tanto al mouse, che consente rapidi e comodissimi scroll semplicemente clickado alle estremità del testo visualizzato, quanto ai tasti cursore. Particolarmente curato questo aspetto, vista l'abbondanza di scelte possibili. Pressando i tasti cursore assieme ad **Alt**, si ottiene infatti un normale scorrimento per singoli caratteri, ma ultraveloce. Con lo **Shift**, ci si può invece spostare da parola a parola nei movimenti orizzontali, e di 4 righe verso o l'alto o verso il basso tramite i corrispondenti tasti cursore. E non è ancora finita: li si può associare anche alla pressione del **Ctrl**, che porta il cursore ad inizio / fine riga, o a inizio / fine documento.

Ce n'è insomma per tutti i gusti: sostenitori dell'uso della tastiera ad oltranza, e amighi che non riescono a tenere lontana la mano dall'adorato mouse. Con C1-Text è solo un problema di scelta.

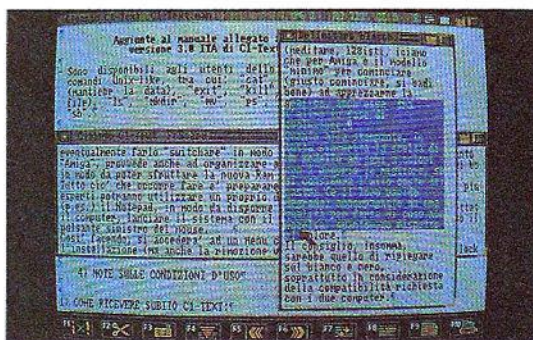
Non solo novità

Data l'enorme quantità di opzioni disponibili, è praticamente impossibile dilungarsi su tutte le doti del C1-Text, alcune delle quali sono tra l'altro state spesso argomento di trattazione su questa rivista. Da citare, per l'ennesima volta, la possibilità di caricare o salvare i testi adoperando **set di caratteri** diversi da quelli di Amiga, il che consente una flessibilità decisamente fuori dal comune per applicazioni riservate alla semplice scrittura, per non parlare dell'ottimo strumento di **conversione** tra diversi standard che si ha a disposizione: da Amiga a **Ms-Dos** (e viceversa), **Atari**, **Macintosh**, e addirittura "vecchi" Commodore come il **C/64** o il **Pet**.

Impossibile non citare ancora, tra quanto "già c'era", la minuziosa ed "attiva" gestione degli **errori**, in grado di intercettare e correggere automaticamente persino una scorretta accenta-

tura, per non parlare del controllo delle maiuscole, delle eventuali ripetizioni, e chi più ne ha più ne metta. Tutti gli errori, o anche quelli solo probabili, vengono segnalati normalmente con un beep accompagnato dal flash dello schermo. Per inciso, anche l'intensità dei beep è controllabile (menu Parametri / Audio), gra zie ad un requester a cursori attraverso il quale si può regolare (o azzerare) il volume tanto dei suoni legati alle segnalazioni di errore, quanto quello collegato a particolari selezioni. Un sistema così completo non poteva non comprendere poi la capacità di adoperare nell'ambito di un documento le cosiddette "etichet te", che consentono, per esempio, di stilare un'unica lettera per tutti i nostri amici, adoperando al posto del nome un'"etichetta" (menu Parametri / carattere).

Un file a parte conterrà poi le corrispondenze con le etichette, in modo che ogni lettera risulterà personalizzata con il nome del destinatario. Ma questa è già una funzione decisamente evoluta, comunque ampiamente descritta nell'indispensabile manuale in dotazione.

Tornando all'editing in senso stretto, non si era accennato a qualche altra chicca, come ad esempio la possibilità di convertire **da minuscolo a maiuscolo** (e viceversa) un intero blocco di caratteri, o anche fargli assumere caratteristiche diverse da quelle impostate in origine: sottolineato, *corsivo*, o **neretto**.

Stessa sovrabbondanza nelle opzioni di stampa, anch'esse indispensabili per definire la qualità di un word processor. Alla disponibilità di settaggi già presenti nelle versioni precedenti, con la 3.0 viene ad aggiungersi una compatibilità di rilievo assolutamente professionale: il **Postscript**. Per tale modalità di stampa, divenuta uno standard di fatto per modelli evoluti di stampanti, il C1-text dispone di un omonimo driver contenuto nella directory principale del dischetto, eventualmente da trasferire assieme al programma se questo viene installato altrove (p. es. un hard disk).

Ma non è questa l'unica novità della release 3.0, anche se le altre possono non essere di immediata ricognizione.

A parte il dichiarato miglioramento "interno" di molte funzioni del text editor, di rilievo la **compatibilità** del programma con la **versione 2.0** del sistema operativo, il che mette al sicuro da sorprese gli eventuali aspiranti al "grande passo" verso un **Amiga 3000**, o comunque intenzionati ad aggiornare il proprio computer quando (e se) questa versione risulterà accessibile ad un normale 500 o 2000.

Nello stesso ambito innovativo, può essere annoverata anche la facoltà di modificare la risoluzione video (menu Parametri / Video). Questa può essere impostata in modo che allo start del programma (dopo aver memorizzato la scelta) lo schermo si adegui allo standard NTSC (250 pixel in verticale), oppure utilizzi delle sezioni extra dello schermo in "overscan", tanto in larghezza che in altezza.

Tra le altre scelte, spiccano poi l'innovativo **Modo Esteso**, in grado di gestire le modalità grafiche legate all'imminente introduzione dell'Enanched Chip Set (**ECS**) che, con adeguati monitor, può supportare il **Vga** standard (tipico degli Ms-Dos), o risoluzioni fino a 1024 x 1008 pixel (scelta 2024, con riferimento al modello di monitor A-2024).

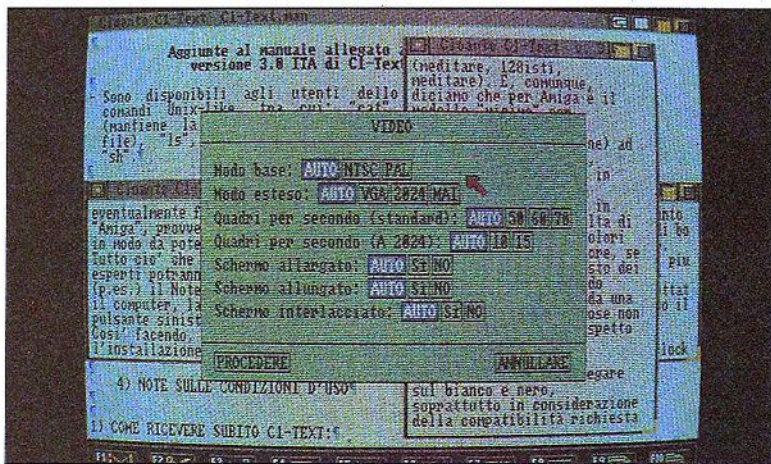
Anche in questo caso, comunque, tutte le selezioni possono essere impostate in modalità **auto**, lasciando al programma il compito di adeguarsi alla configurazione grafica e di memoria di cui si dispone.

Va infatti ricordato che, nonstante la mole e la qualità delle opzioni accessibili, C1-Text è in grado di "girare" anche in configurazione minima, ovvero con un **Amiga 500 non espanso**.

Chiaro che, in simili condizioni, qualche restrizione risulterà d'obbligo: il programma cercherà di eliminare il Workbench, e documenti lunghi oltre una certa mole (più o meno oltre i 100 Kbyte, niente paura) non risulteranno gestibili correttamente. Ma Amiga con soli 512 K, ormai, sono praticamente un'eccezione.

Per concludere non resta che accennare all'esiguo costo del software, almeno in rapporto a prodotti del genere, ed alla possibilità, per chi già è in possesso di una versione antecedente del programma, di **avere la nuova release ad un prezzo praticamente simbolico.**

Vogliamo affibbiare un voto a C1-Text 3.0?

Presto fatto: non sarà un 30 e lode (per migliorare c'è sempre tempo...), ma un 30 non glielo toglie nessuno.

a cura di Goran Fisher

MOVIE SETTER "PARLA" ITALIANO

Il famoso programma di animazione per Amiga viene finalmente commercializzato anche in lingua italiana

Comic Setter è un famoso prodotto della **Gold Disk** recentemente tradotto completamente in italiano, sia per quanto riguarda la **manualistica**, sia per quanto riguarda **i codici dei programmi**. Si tratta di un generatore di animazioni di qualità professionale, estremamente semplice ed intuitivo da usare, che ha riscosso un vasto successo in tutto il mondo.

Essendo correntemente distribuiti in Italia da una grossa casa, la LEADER di Casciago, è facilmente reperibile in molti computer shop sparsi per la nostra penisola.

Il pacchetto

Movie Setter viene fornito su due dischi, uno contenente il programma, l'altro dei files esemplificativi e di supporto, con un manuale che è la edizione italiana, riveduta e corretta (ove necessario) dell'originale.

La traduzione del programma è stata svolta sui tabulati di testi forniti dalla Gold Disk da parte di **Luigi Callegari**, da anni collaboratore delle nostra testata e profondo conoscitore del "sistema" Amiga. Sua è anche la **traduzione** del manuale, che ovviamente contempla tutti i riferimenti nominali (**funzioni** dei menu, **gadget**, eccetera) così come sono presenti nella edizione italiana tradotta del software su disco. Il manuale, di circa settanta pagine, è stato revisionato ed impaginato da un gruppo di specialisti, in modo da garantire un prodotto di qualità almeno pari all'originale, conservando la varietà di figure, le didascalie e gli stili del manuale originale della Gold Disk.

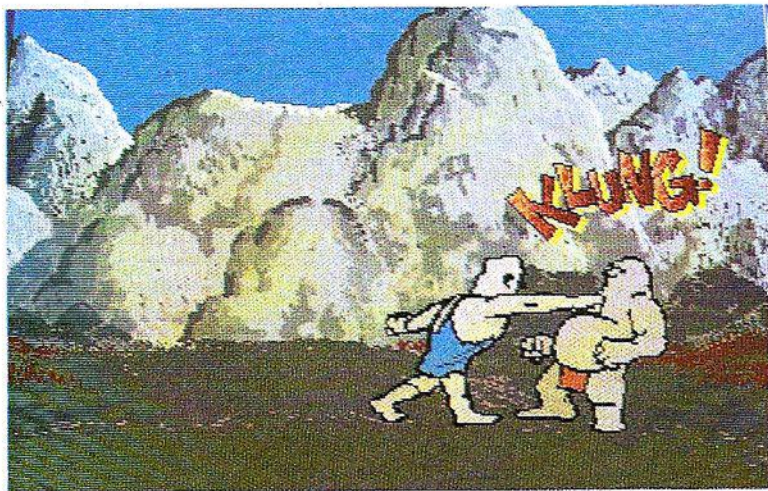
Installazione

Il programma funziona con qualunque tipo di **Amiga in ogni configurazione** a partire da quella di base (perfino A-500 con 512K di RAM ed il solo drive interno) sebbene il manuale caldeggi una configurazione con almeno 1024 K di memoria per potere, se non altro, realizzare animazioni di lunghezza significativa.

Se si deve installare Movie Setter su **HD** è sufficiente preparare un cassettino di icona ed una directory per inserirvi i files che compongono il programma, che infatti viene caricato "a moduli" per evitare inutili sprechi di memoria. Prima di agire, il programma richiede una parola italiana presente sul manuale originale. E' solo una piccola **protezione** contro la copiatura indiscriminata di un pacchetto che vale certo i soldi che costa.

Caratteristiche

Il principio su cui si basa Movie Setter per funzionare è il classico **"page flipping"**, ovvero lo scorrimento ad alta ve-

locità sul video di pagine grafiche preparate in memoria, ciascuna contenente un fotogramma (**frame**, in inglese) della sequenza di animazione, proprio come in un cartone animato. Si tratta di una tecnica non particolarmente evoluta, specie se raffrontata con alcuni pacchetti (come **Zoetrope**) che consentono addirittura animazioni tridimensionali, ma comunque perfettamente in grado di svolgere gli scopi del programma.

I meriti principali del programma Movie Setter sono altri: la sua intuitività e la possibilità di preparare, in tempo reale, un'animazione secondo la modalità *Ciò che vedi è quello che otterrai davvero* (in inglese si sigla con l'acronimo **WYSIWYG**, = What You See Is What You Get). Movie Setter svolge da sè tutte le funzioni di gestione dell'animazione, consentendo all'utente di concentrarsi solo sui particolari, per favorire l'immediata visione della sequenza progettata.

Come detto, Movie Setter è suddiviso in più files, che rappresentano fisicamente due moduli: il **Set Editor** e lo **Scene Editor**. Quando si vuole costruire o modificare un insieme di particolari (un "set", appunto) componenti un fotogramma dell'animazione, si usa il primo programma; altrimenti, con il secondo, si lavora globalmente sulla sequenza di set che compongono l'animazione. Se si possiede almeno un megabyte di memoria, il programma può eventualmente caricare ambedue i moduli in memoria per consentire la commutazione immediata da uno all'altro durante il lavoro.

Alcuni vocaboli

Per capire il funzionamento di Comic Setter, come di altri programmi di animazione che lavorano sugli stessi principi, è bene precisare il significato di alcune parole inglesi praticamente "intraducibili".

Con **Set**, letteralmente "insieme", si intende una sequenza di immagini (chiamate in inglese **Faces**, facce) che, visualizzate rapidamente una dopo l'altra, riproducono l'animazione. Il **Background** è lo "sfondo", solitamente una immagine **IFF** che rimane costantemente visualizzata o, al più, fatta scorrere dietro ad alcuni protagonisti, grazie ad apposite funzioni del programma.

Una Traccia, in inglese **Track**, è una serie di immagini posizionata sullo sfondo, mentre un Evento, in inglese **Event**, è una definizione di un gruppo di caratteristiche che devono essere cambiate o generate in un certo momento (fase) dell'animazione; ad esempio: suoni, colori di sfondo, eccetera. Un **Wipe** è un effetto di comparsa dello sfondo nell'animazione, in sfumatura oppure per bande orizzontali.

Il manualetto in dotazione riporta, comunque, chiari esempi di applicazioni pratiche.

Usando Movie Setter

Volendo costruire una prima, semplice animazione, si deve innanzitutto stabilire lo sfondo, il **Background**. Eventualmente, per semplicità, possiamo inizialmente definire uno sfondo nullo, opzionando **Vuoto**. Dopo avere selezionato, da requester, lo sfondo desiderato (eventualmente caricando un'immagine IFF tra quelle fornite sul secondo disco nella apposita directory **Background**), appare una finestra con **sei** tipi diversi di effetti **Wipes** per gli sfondi. Non volendo usare inizialmente alcun effetto, possiamo selezionare **No**.

Per il set da utilizzare si sceglie quindi **Nuovo** dal menu **Traccia**, caricando magari dal dischetto di programma una serie di immagini preconfezionate da disporre sullo schermo (un marzianino, una donna, una sfera, eccetera). Accanto al puntatore del mouse abbiamo la prima immagine del set: è sufficiente quindi cliccare la posizione in cui vogliamo la prima immagine per terminare il lavoro con il primo fotogramma. Automaticamente Movie Setter ci colloca nel secondo frame, come si può riscontrare dall'apposito indicatore nella barra del menu, mentre il puntatore del mouse mostra la seconda immagine.

Se, ad esempio, stiamo producendo l'animazione di un omino che cammina da **destra** verso **sinistra**, basterà collocare il puntatore del mouse con la sua immagine un po' a sinistra della precedente e cliccare ancora. Si prosegue, così, sino a che non si posiziona l'ultima immagine; quindi si preme il tasto **Ctrl** insieme al pulsante sinistro del mouse. Non vi è un limite al numero di tracce visualizzabili nello stesso tempo, se non la memoria disponibile.

Dopo avere collocato l'ultima immagine, si accede ad un requester con vari gadget per redigere lo scheletro di animazione preparato. E' ovviamente possibile **ritagliare** e **copiare** porzioni di immagini, **avanzare** e **retrocedere** nel set, nonchè agire sulla **priorità** di visualizza-

Programmi italiani

Attualmente, per **Amiga**, non esistono molti programmi in italiano e sono pochi anche quelli con il solo manuale tradotto dall'inglese. I motivi sono parecchi e di vario tipo.

Innanzitutto bisogna reperire (e compensare) persone esperte nel campo specifico (altrimenti ci si ritrova con traduzioni letterali, spesso ridicole ed incomprensibili); occorrono, inoltre, un impaginatore che riproduca con sufficiente qualità il manuale originale; un tipografo, ed una s/w house che si sobbarchi l'onere finanziario.

Tutte le spese, e lo sforzo organizzativo, sono quasi sempre a carico di un piccolo distributore, in quanto solo poche softhouse (Microsoft o Borland, ad esempio) dispongono di filiali italiane preposte allo sviluppo di prodotti nazionalizzati. Affinchè una piccola softhouse decida di produrre software nazionalizzato, si deve presumere che il prodotto venga venduto in numero sufficiente di esemplari, in modo da ricoprire le spese sostenute. Dal momento che Amiga ha, tuttora, un mercato ristretto, in particolare per quanto riguarda gli applicativi professionali, e per giunta afflitto dal problema della pirateria (che lima ulteriormente gli utili) si capisce bene come mai sono così pochi i pacchetti tradotti.

Per quanto riguarda, poi, la traduzione dei pacchetti software, bisogna dire che il problema diventa arduo quando è necessario tradurre anche il programma. Infatti chi è incaricato della traduzione svolge il compito operando su tabulati, o files Ascii contenenti le parole presenti nel programma. Il traduttore, quindi, non solo deve individuare il "luogo" ove sono inserite le frasi, ma anche effettauare traduzioni comprensibili; conservando, ovviamente, la stessa lunghezza del testo originale. Se, ad esempio, un requester presenta il messaggio **Box** (di 3 caratteri), delimitato da un contorno, se si traduce il termine con "Rettangolo" (di 10 caratteri) si dovrebbe riprogrammare la parte di tracciatura del gadget, ammesso che sia possibile, date le dimensioni del requester sullo schermo. In questi casi, solitamente, il traduttore deve ricorrere ad abbreviazioni (la lingua inglese è, per giunta, solitamente più "corta" e sintetica dell'italiano) e spesso si rischiano dei compromessi del tutto inintellegibili.

Capita spesso, paradossalmente, che anche utenti esperti non si raccapezzano più sui termini tradotti. Un programmatore, ad esempio, sa perfettamente che cosa significa "Parsing", ma se nel pacchetto di un assemblatore nazionalizzato trova "Scansione" potrebbe non capire esattamente che cosa significa....

zione di vari oggetti (ad esempio, per fare sparire il nostro ometto dietro un muro mentre sta camminando). Tramite la funzione **Muovi Traccia** è addirittura possibile spostare, via mouse, un'immagine o l'intero set in un colpo solo.

Sempre nel menu Traccia, troviamo un'altra opzione molto utile: **Tieni** ripete, per vari frames, l'ultima posizione di una immagine indicata via mouse (senza ripetere, ovviamente, tutto il contenuto del frame, nel qual caso dobbiamo appunto impartire il comando **Duplica** del menu Traccia).

Gli eventi

Sinora abbiamo capito come sia possibile produrre, con Movie Setter, animazioni consistenti in semplici sequenze di immagini mosse sullo schermo. Per ravvivare la cosa si possono inserire particolari "esterni" come, ad esempio, il rumore del contatto col suolo nella scena di una sfera che rimbalza; oppure, se una valanga scende da una montagna, lo scorrimento dello sfondo, eccetera.

Per questo scopo è previsto il menu **Evento**, contenente una serie di funzioni molto evolute per introdurre particolari di ogni tipo nelle animazioni. Per ogni evento viene prodotto, sullo schermo, un requester specifico.

Ad esempio, volendo introdurre un suono, appare una finestra con una tastiera musicale ed un gadget che consente di scegliere l'ottava.

Ovviamente le capacità sonore sfruttano l'effetto stereo ed agiscono in pieno multitasking durante l'animazione.

Si possono settare **cicli di colori** per produrre "false" animazioni sfruttando l'alternarsi di colori (ad esempio in un bel falò); lo scorrimento dello sfondo via Blitter, oppure la modifica globale della tavolozza (**Palette**) dei colori usati nell'animazione. Questi risultano, per ogni fotogramma, in numero massimo di 32, da scegliere nella tavolozza dei 4096 disponibili su Amiga.

Conclusioni

Descrivere in modo esauriente un programma complesso come Movie Setter, pur se intuitivo da usare, avrebbe richiesto moltissimo spazio.

Queste pagine, pertanto, hanno il solo scopo di far comprendere l'estrema versatilità e semplicità d'uso del prodotto. Movie Setter è pertanto consigliabile a chiunque voglia immedesimarsi nei panni di Walt Disney, acquistando il pacchetto originale completo di manuale italiano.

di Luigi Callegari

I COMPILATORI "C" PER AMIGA

Come promesso in precedenza, ecco che inizia la grande avventura del "C" su Amiga; iniziando a parlare, appunto, proprio dei compilatori disponibili

Il C è il linguaggio di sviluppo principe per Amiga. Oltre il 90% del software commerciale e di pubblico dominio circolante è stato scritto, quasi sempre in toto, sfruttando questo linguaggio.

Persino la documentazione esistente, a partire da quella originale della Commodore riportata nei famosi **Rom Kernel Manuals**, è stata scritta proponendo esempi quasi esclusivamente in C e le librerie interne di Amiga sono tali da interfacciarsi agevolmente con il compilatore.

Per Amiga esistono almeno tre compilatori di pubblico dominio e due pacchetti commerciali di sviluppo di applicativi in C; si tratta del **Lattice C Compiler** prodotto dalla Lattice / SAS (giunto alla versione **5.10**) e dello **Aztec C Compiler** prodotto dalla Manx Inc. (giunto alla versione **5.0b**), dei quali descriveremo le principali differenze, allo scopo di guidare gli interessati all'acquisto più appropriato.

I compilatori di Amiga sono tutti del tipo **Ansi C**. Ciò significa che, almeno per quanto riguarda le nozioni di base, tutto quello che viene appreso può essere integralmente "trasferito" in altri ambienti di lavoro, come ad esempio **Ms - Dos** e **Unix**.

Ovviamente il discorso non vale quando si parla dell'uso delle **librerie interne** di Amiga, che sono peculiari del nostro beniamino.

Problemi di lingua

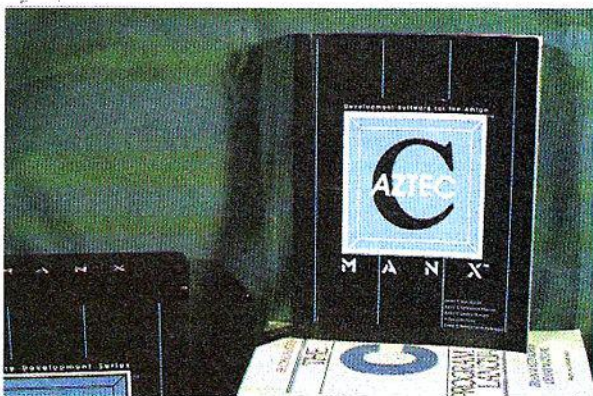
Lattice C ed Aztec C sono pacchetti disponibili **soltanto** in lingua inglese. Del resto chi vuole studiare seriamente la programmazione di Amiga deve rassegnarsi: o impara l'inglese, o si ritroverà presto con il fiato corto. Quasi tutta la documentazione di valore e tutto il software di pubblico dominio (anche didattico) per Amiga è infatti scritta in lingua inglese (ma non spaventatevi: talvolta si trovano buone trattazioni anche in tedesco...).

Quindi è necessario mettersi in testa che, almeno sinchè Amiga non avrà un mercato sufficientemente ampio e professionale (da giustificare l'italianizzazione di pacchetti applicativi, come già è avvenuto per il mondo Ms-Dos), dovremo necessariamente sforzarci di capire l'inglese tecnico, che non è poi così difficile, almeno se ci si appassiona.

Per quanto riguarda l'acquisto dei pacchetti originali, atto necessario per disporre dell'indispensabile documentazione e di eventuali upgrades dei pacchetti, ci si può rivolgere ai negozi (super)specializzati nell'importazione di software originale, come ad esempio la **Alex Computer** di Torino o la **Digimail** di Milano. Alternativamente si può pensare di acquistare negli USA, magari tramite carta di credito, presso uno dei tanti negozi che effettuano vendite per corrispondenza. Il prezzo di acquisto è compreso tra i 150 ed i 200 dollari statunitensi.

I pacchetti

Lattice C ed Aztec C vengono forniti, rispettivamente, su **cinque** e **quattro** dischetti, corredati di corposi manuali d'uso.

Ambedue possono teoricamente funzionare su di un Amiga con **almeno 1024**

Opzioni di Lattice C

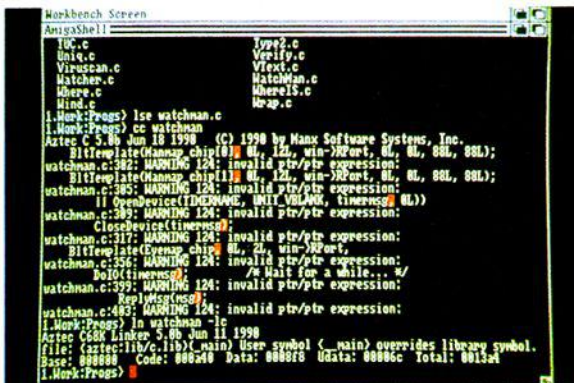
Ecco l'elenco riassuntivo delle opzioni di compilazione di **Lattice C V5**. Si ricordi che possono essere specificate quante opzioni si vogliono su di una stessa linea di comando, compatibilmente con la massima lunghezza della linea digitabile da Shell. Ogni opzione, o serie di opzioni, inizia con il segno "meno" (-) seguito da quante opzioni "singola lettera" si vogliono.

Se, invece, si devono specificare **switch** di compilazione che iniziano tutti con la stessa lettera (ad esempio: **-ca**, **-cf**, **-ct** ecc.), dopo ogni "meno" deve esserci la lettera di inizio switch seguita da tutte le lettere desiderate.

Esempio: **-caft** ecc..

Opzione	Descrizione
ab	Pone il blocco BSS in memoria CHIP
ac	Pone il blocco Code in memoria CHIP
ad	Pone il blocco Data in memoria CHIP
b0	Usa indirizzamento dati non Relativo alla Base
b1	Usa indirizzamento dati Relativo alla Base
c+	Sopprime warning per uso improprio di strutture
ca	Garantisce compatibilità ANSI
cc	Consente annidamento dei commenti
cd	Consente di usare $ nbegli indetificatori
ce	Sopprime la stampa della linea errata
cf	Richiede prototipi di ogni funzione
ci	Sopprime inclusioni multiple
ck	Consente l'uso di nuove keywords
cm	Consente l'uso di costanti alabnum. multiple
co	Abilita preprocessore vecchio stile
cr	Consente il parsing degli argomenti in registri
cs	Crea una sola copia per costanti stringa doppie
ct	Genera warning per tags usati senza definizione
cu	Forza i char in unsigned char
cw	Disabilita warning per funz. void non dichiarate
cx	Tratta tutte le dichiaraz. globali come esterne
c	Continua in caso di errore
d0	Disabilita il debugging
d1	Abilita il debugging semplice
d2	Genera informazioni simboliche
d3	Genera informazioni simboliche e dump ogni linea
d4	Genera informazioni simboliche complete
d5	Genera informaz. simboliche compete ogni linea
ff	Usa matematica in virgola mobile FFP
f8	Genera codice diretto per coprocess. matematici
fi	Genera codice per librerie IEEE
fl	Genera codice per librerie Lattice
gc	Simboli in definizione incrociata negli include
gd	Simboli in definizione incrociata
ge	Lista le linee escluse
gh	Lista i files di header
gi	Lista i files di inclusione
gm	Lista le espansioni di macrodefinizioni
gn	Stampa linee più compatte
gs	Lista il sorgente
hb	Carica blocco BSS in FAST RAM
hc	Carica blocco Code in FAST RAM
hd	Carica blocco Data in FAST RAM
ix	Specifica la directory di inclusione
jn	Disabilita il messaggio di avvertimento "n"
L	Chiama il linker
l	Allinea oggetti a dimensione di longword
M	Compila solo sorgenti modificati di recente
m0	Genera codice per processore 68000
m1	Genera codice per processore 68010
m2	Genera codice per processore 68020
m3	Genera codice per processore 68030
ma	Genera codice per qualunque processore
mc	Disabilita ottimizzazione di cleanup
mr	Disabilita registrazione di parametri
ms	Ottimizza per spazio
mt	Ottimizza per velocità
n	Conserva solo otto caratteri per identificatore
O	Chiama l'ottimizzatore globale di codice GO
ox	Scrive in "x" il codice compilato
p	Genera istruzioni di stack probe
pe	Genera prototipi per funzioni esterne
ph	Genera header di files precompilati
pp	Genera prototipi con _PARMS per trasferibilità
pr	genera file di prototipi
ps	Genera prototipi solo per funzioni statiche
qx	Pone il quad file in "x"
r0	Chiamate subroutine assolute
r1	Chiamate subroutine PC-relativo
rr	Usa parametri registrati per chiamate subroutine
rs	Usa parametri su stack per chiamate subroutine
rb	Usa indifferentem. param. registrati e su stack
s	Specifica i nomi di segmenti
u	Elimina definizioni di tutti i simboli standard
v	Disabilita generaz. codice di controllo stack
w	Usa INTeri a sedici bit anzichè a 32 bit
x	Tratta tutte le dichiaraz. globali come esterne
y	Carica A4 con l'indirizzo di base

K di memoria e due drives. Chi possiede più memoria e, soprattutto, un disco rigido, potrà lavorare con efficienza enormemente superiore.

Per quanto riguarda i possessori di sistemi dotati dei soli floppy, bisogna dire che Aztec C richiede meno memoria per funzionare, in quanto i suoi eseguibili (compilatore CC, assemblatore AS e linker LN) sono più compatti. Aztec C prevede inoltre un programma che esegue automaticamente l'installazione del sistema di sviluppo su di una serie di dischetti di lavóro eseguendo copie di backup già configurate adeguatamente, oppure su Hard Disk. Per evitare di saturare i dischetti di lavoro è consentito scegliere inizialmente con quali librerie matematiche si desidera lavorare: **FFP** (Matematica in virgola mobile e singola precisione), **IEEE** (Matematica in virgola mobile in doppia precisione) o **Aztec** (in virgola mobi-

Opzioni di compilazione di Aztec C V5.0

Ecco le opzioni di compilazione dell'Aztec C V5. Si badi che la sintassi è completamente cambiata rispetto alla precedente versione 3.6a, con la quale però è stata mantenuta la completa compatibilità tramite lo switch **-3**, che consente di specificare le opzioni di compilazione nel vecchio standard non ANSI.

Opzione	Descrizione
3	Opzioni seguenti secondo lo standard 3.6
5	Opzioni seguenti secondo lo standard 5.0
a	Non assembla al termine della compilazione
at	Come a, ma commentando il listato
bd	Abilita il controllo dei tipi
bs	Produce infomazioni per il debugger
c2	Compila per processori 68020/030
d	Definisce un simbolo per il preprocessore
fa	Genera codice matematica IEEE
ff	Genera codice matematica FFP
fm	Genera codice matematica Manx IEEE
f8	Genera codice matematica per FPU
hi	Legge header precompilati
ho	Scrive tabella simbolica in file precompilato
i	Specifica la path dei files di inclusione
k	Compila codice secondo le regole K&R (Unix 7)
ma	Forza l'allineamento di short e long
mb	Genera la specifica .begin
mc	Genera codice utilizzante il modello large code
md	Genera codice utilizzante il modello large data
me	Allinea stringhe a dimensione parola
mm	Pone il codice nel segmento dati
ms	Pone stringhe statiche nel segmento dati
o	Specifica il nome del file di output
pa	Abilita preprocessore ANSI e trigrafici
pb	Usa unsigned come default dei bitfield
pl	Usa interi a 32 bit anzichè a 16
po	Usa preprocessore versione 3.6
pp	Usa unsigned come default
ps	Definisce interi a 16 bit invece che a 32
pt	Cerca trigrafici nello stream di input
pu	Usa le regole di conservazione usigned
qa	Causa la generazione di prototipi con _PARMS
qf	Abilita QuickFix
qp	Genera protitpi per tutte le funzioni non statiche
qq	Disabilita messaggi di startup e errore
qs	Genera prototipi per le funzioni statiche
qv	Genera informazioni verbose sull'uso di memoria
r4	Usa A4 come variabile registro
sa	Abilita compilazione in due passate per ottimizzazione
sb	Genera codice inline per funzioni stringa
sf	Ottimizza i cicli for (;;) spostando il test
sn	Non genera LINK e UNLK se non necessari
so	Usa tutte le ottimizzazioni possibili
sp	Ritarda il popping degli argomenti sinchè possibile
sr	Registra i parametri in base all'uso
ss	Usa un solo puntatore per costanti stringa identiche
su	Alloca registri secondo uso, ma rispettando precedenze
wa	Verifica prototipi con parametri passati alle funzioni
wd	Genera messaggi di warning per definizioni K&R
we	Tratta i warning come errori
wl	Eguale a -waru
wn	Non generea warning per conversioni puntatore-puntatore
wo	Solo warning per conversione intero-puntatore
wp	Genra un warning per funzioni senza prototipi
wq	Invia i warnings al file Aztecc.err
ws	Ignora tutti i warning
wu	Avverte se vi sono variabili locali inutilizzate
ww	Prosegue comunque anche dopo cinque errori

le e doppia precisione). Per quanto riguarda Lattice, è quasi indispensabile inserire tutti i suoi numerosi files eseguibili di supporto (molto utili durante lo sviluppo del programma) su di un disco rigido per richiamarli in fretta senza troppi scambi di dischetti. Oppure si deve disporre di un bel po' di Ram da configurare come **Ram Disk**.

Si tenga conto che sviluppare programmi in C consiste in ripetuti processi di digitazione, compilazione, linking e prova; il tutto da ripetere sempre integralmente, a differenza di quanto avviene con linguaggi interpretati, come il Basic. Ciò significa che devono trascorrere parecchi secondi per ogni compilazione, attesa del caricamento di files di inclusione da disco, codificazione vera e propria da parte del compiler, linking ed altri ancora per testare il programma. Anche lo sviluppo di programmi poco più che banali richiede, pertanto, una mole di tempo intollerabile se si dipone di poca Ram e si è privi del disco rigido.

Alcune differenze

Lattice C viene fornito come due files eseguibili (**LC1** e **LC2**) che svolgono le due fasi di compilazione necessarie per tradurre un file sorgente Ascii in un codice oggetto. Il **Linker BLink** consente poi di legare il programma con le necessarie funzioni di supporto e di interfacciamento per le librerie di Amiga, in modo da ottenere un programma eseguibile effettivamente da Shell o da Workbench.

Sono presenti anche: un **debugger SDB**, che consente di testare il programma alla ricerca di bug; un **ottimizzatore** globale di codice **GO**; una versione speciale del compilatore **LC1b**, che consente funzioni supplementari; un **editor** di schermo **LSE** studiato dalla Lattice per creare un ambiente di sviluppo software integrato (simile a quello dei Turbo C e QuickC del mondo Ms-dos); ed altri programmi di supporto.

Aztec C è costituito da un compilatore C che genera, come risultato (partendo da un testo sorgente Ascii) un listato **assembler standard metacomco** in Ascii. La seconda "passata" di compilazione è svolta da **AS** che è null'altro che un assemblatore standard Motorola (con varie peculiarità). Infine il **linker LN** svolge la fusione del codice assemblato con le librerie di supporto, che sono in un formato particolare della **Manx** e non nel formato standard universale "**.lib**" come per **BLink**.

Lattice C è più **ricco** di opzioni, Aztec C è più **agile** e veloce. Il primo è più pignolo nella verifica del testo sorgente ed è quindi probabilmente più adatto ai principianti che, altrimenti, si trovano subito a lottare contro formidabili errori nel sorgente, del tutto ignorati dal compilato-

Ai lettori

Inutile dire che, da questo momento, viene "aperto" il colloquio diretto con i nostri lettori che possono quindi rivolgerci domande, proporre casi particolari, chiedere delucidazioni su argomenti specifici. Una rubrica fissa sul "C" verrà pertanto sviluppata e si affiancherà, spesso "fondendosi", con quella già attivata con il mondo Ms-Dos; non dimentichiamo,infatti, che una delle caratteristiche più importanti di un qualsiasi compilatore **C** è la "trasportabilità" dei programmi sorgenti, almeno finchè non vengono interessate librerie specifiche di un particolare computer.

re. Il fatto che Aztec C generi, come codice intermedio, del puro Assembler, costituisce per molti esperti di programmazione in linguaggio macchina un motivo preferenziale nella scelta, oltre alla minore richieste di spazio su dischetti.

Uso pratico

Per dare un'idea pratica di come si produce un semplice programma in C con i compilatori commerciali di Amiga, abbiamo preparato un breve listato assembler elementare in C, riportato nello specifico riquadro.

Digitatelo con un qualunque editor Ascii e registratelo su disco. Lattice C prevede un editor interno, **LSE**, che può essere usato per lo scopo. Aztec C non prevede un editor, quindi si deve usare uno dei tanti editor di pubblico dominio o commerciali noti: Memacs (fornito dalla Commodore, con tutti gli Amiga), Ed, Cygnus Ed, eccetera. Teoricamente è possibile usare anche dei word processor, purchè consentano di scrivere il file in formato Ascii puro, privo cioè di codici di controllo. Generalmente è meglio usare editor di linea come quelli citati che, essendo stati sviluppati espressamente per questo compito, risultano certamente più pratici da usare.

Dunque, supponendo di usare Lattice C lanceremo...

```
LSE testo.c
```

...digiteremo il listato e usciremo con l'apposita funzione **Save and Exit**, che scriverà quanto da noi digitato nel file dal nome, appunto, di "**testo.c**". Questo viene sempre scritto , salvo apposita ridenominazione, nella "directory corrente", fissata con **CD**. Spesso si usa scrivere un listato sorgente nella **Ram Disk**, per esempio con:

```
Emacs RAM:Testo.c
che è equivalente a:
CD RAM:
Emacs Testo.c
```

In ogni caso, quando poi si invocherà il compilatore, è necessario specificare la directory di compilazione. Se si è già assegnata la directory corrente con CD, anche i files oggetto ed eseguibile saranno in questa scritti, per default, dal compilatore e dal linker.

Ad esempio digitando, con Lattice C...

```
LC -Lm -fi Testo.c
```

...il comando invocherà il compilatore **LC** (che a sua volta chiama **LC1** e **LC2**), specificando di compilare il file di nome **testo.c** con l'opzione **-Lm**; questa richiama il linker imponendo di linkare con le librerie matematiche (dal momento che il programma usa funzioni matematiche), mentre il parametro **-fi** indica al compilatore di generare il codice che usa la libreria matematica IEEE in doppia precisione.

In Aztec C dovremo, invece, usare due linee di comando:

```
CC -fa Testo.c
LN Testo c.lib ma.lib
```

La prima linea impone di compilare ed assemblare il sorgente **Testo.c** (con cancellazione del file intermedio in assembler per default) usando le librerie standard Commodore IEEE. La seconda linea esegue il linking del modulo **testo.o** con la libreria **c.lib** (contenente le funzioni **Manx**) e con la libreria **ma.lib** (contenente il codice di interfaccia per le librerie matematiche Commodore IEEE). Alla fine, per eseguire il codice eseguibile si dovrà digitare, da Shell, semplicemente: **testo** (e return) per vedere funzionare il programma.

```
/* Soluzione di equazioni di secondo grado reali
   Versione ANSI C per Lattice C ed Aztec C V5.0
*/
#include <stdio.h>
#include <math.h>
void main( int argc, char *argv[] )
{
   float a, b, c, d, x1, x2;

   printf( "Soluzione di: aX2 + bX + c = 0\n" );

   printf("Immetti a \n");
   scanf( "%f", &a );

   printf("Immetti b \n");
   scanf( "%f", &b );

   printf("Immetti c \n");
   scanf( "%f", &c );
   d = b * b - 4.0 * a * c;

   if ( d < 0 ) {
      printf( "Radici immaginarie: delta < 0!\n");
      exit( 0L );
   };

   x1 = ( -b - sqrt( d )) / ( 2 * a );
   x2 = ( -b + sqrt( d )) / ( 2 * a );

   printf( " Radice 1 = %f\n", x1 );
   printf( " Radice 2 = %f\n", x2 );
}
```

di Giancarlo Mariani

UN MICROCAD VERSIONE "C"

Forse il listato che presentiamo può apparire un po' lungo. Rappresenta, tuttavia, la "conversione" di un listato Basic pubblicato tempo addietro

Il listato proposto in queste pagine non è altro che il programma **Microcad** (proposto su C.C.C. n. 78 nella versione **Gw-Basic**), ma convertito nel linguaggio **Borland's Turbo - C 2.0**.

MicroCad, per chi non lo ricordasse, è utile per disegnare, ma, come il precedente listato Gw Basic, si propone di essere un programma didattico: di cose da imparare, infatti, ve ne sono una miriade. Potrete dare uno sguardo alla **gestione dei files sequenziali**, alla **grafica**, alla **gestione di tabelle**, ma soprattutto potrete imparare a conoscere meglio il compilatore **C** ed a raffrontare le sue caratteristiche con quelle del più antiquato Basic.

In realtà il programma, essendo una traduzione dal Basic, utilizza solo in parte le potenti strutture del C, ma può servire ugualmente per cominciare ad apprendere il nuovo linguaggio per avvicinarsi, magari, al mondo Ms-Dos.

Ricorderemo brevemente, per chi non possiede il numero citato, alcune delle caratteristiche del programma originale **Autocad**, potentissimo programma grafico usato professionalmente dai progettisti elettronici ed ingegneri, che possiede un'infinità di funzioni, tramite le quali è possibile fare praticamente ciò che si vuole, come rotazioni 3D, zoom, sovrapposizioni, ed altro.

La caratteristica più importante di AutoCad è che quando registra un disegno su disco, NON registra il disegno vero e proprio (cioè la schermata grafica), ma le istruzioni per comporlo, vale a dire l'intera sequenza di istruzioni tipo "Linee", "Cerchi", "Punti", e tutte le altre che servono per produrre il disegno.

In pratica, ciò che viene registrato da AutoCad è un vero e proprio programma, che, al momento della successiva lettura, viene **interpretato ed eseguito** dal Cad.

Anche MicroCad permette di disegnare linee, rettangoli e cerchi, quindi di memorizzare le istruzioni in un **database**, che può essere modificato a piacimento e quindi registrato o ricaricato da disco.

Per l'uso di Microcad rinviamo il lettore al numero di citato C.C.C.

Qui, infatti, ci limiteremo a descrivere le principali sezioni in cui è suddivisa la versione C del programma.

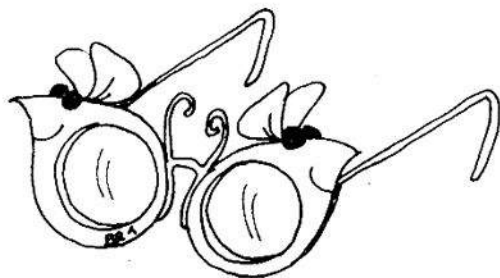
Descrizione del listato

Il listato C si divide in varie parti funzionali, interconnesse tra loro.

Dichiarazione files di include.

Viene effettuata tramite la direttiva del compilatore **#include** seguita dal *nome* di file. Questi files (generalmente con estensione **.H**) vengono praticamente inclusi, ossia caricati da disco ed inseriti nel programma sorgente; contengono tutte le *intestazioni* delle procedure standard del C che permettono al compilatore di eseguire il controllo sul corretto passaggio dei parametri alle sue istruzioni.

Se i files vengono omessi, il programma funziona ugualmente, ma il compilatore produrrà una serie di **Warnings** (messaggi di avvertimento) e non potrà più controllare il corretto passaggio di parametri alle varie istruzioni standard.

Dichiarazione variabili globali

Queste sono le variabili generali, ossia quelle "viste" sia dal **Main** (programma principale) che da tutte le altre procedure e funzioni presenti nel programma. Il numero di queste variabili dovrebbe essere ridotto al minimo, perchè ogni procedura e funzione dovrebbe racchiudere, in sè, tutto ciò che occorre per funzionare; eventuali variabili esterne dovrebbero essere passate come *parametri*.

Dichiarazione prototipi di funzione

Queste sono le intestazioni delle procedure e funzioni sviluppate all'interno del programma, e servono, come i files .H, a far eseguire, al compilatore, il controllo sul passaggio dei parametri, questa volta non alle istruzioni standard C, ma alle procedure e funzioni sviluppate dal programmatore.

Se non ci fossero le intestazioni, il compilatore non potrebbe accorgersi se, ad esempio, ad una procedura passiamo un numero intero al posto di una stringa (o viceversa).

Le procedure

Descriveremo, una per una, tutte le procedure vere e proprie usate nel listato, dichiarate all'inizio grazie al termine **Void**.

SelectType

Scrive su video la maschera di scelta ("Tutto", "Linee", "Rettangoli", "Cerchi") e richiede il numero corrispondente all'opzione desiderata. Se il parametro **inizio** vale 0, sarà permesso scegliere anche "Tutto", altrimenti solo le ultime 3 opzioni saranno possibili. Il parametro **Opz** riporta, in "uscita" dalla procedura, la scelta effettuata. L'asterisco (*) posto davanti al nome della variabile indica che il parametro è passato come "puntatore", altrimenti non si riuscirebbe a modificarne il valore all'interno della procedura. Il parametro Stringa contiene il titolo della maschera, che verrà scritto in alto sullo schermo.

La procedura contiene un ciclo **While**, che si ripete sino all'inserimento di una scelta corretta tra quelle disponibili.

La procedura riproduce la parte del programma Basic (C.C.C. n. 78) numerata dalla linea 9400 alla linea 9570.

CaricaDisegno

Predispone il caricamento disegno da disco e richiama **SelectType** per scegliere il tipo di figura da caricare; quindi, dopo aver richiesto tramite **Scanf** il nome del disegno, richiama la procedura **Carica** che, a seconda della scelta effettuata, carica linee, rettangoli o cerchi.

Carica

Carica il disegno da disco. Il parametro **Tipo** indica il tipo di figura da caricare. Se tipo = 1 verranno caricate le linee, con tipo = 2 i rettangoli e con tipo = 3 i cerchi. **Name** è il nome del file da caricare (passato dalla procedura **CaricaDisegno**), al quale sarà aggiunta la giusta estensione a seconda di tipo.

L'istruzione **Fopen** apre il file, che verrà identificato dalla variabile **Stream**. Il ciclo While (fscanf...) carica un parametro dal file aperto, e controlla che non vi sia una condizione di **EndOfFile**. I 4 parametri vengono caricati, tramite 4 fscanf nelle variabili stringa X1s, X2s, X3s, X4s, e con istruzioni del tipo **atoi** vengono convertiti in interi ed immessi nell'array temporaneo **Temp.** Alla fine del caricamento, tramite la procedura **Temp2Arr**, l'array Temp viene travasato nell'array voluto delle Linee, dei Rettangoli o dei Cerchi.

Il caricamento del disegno era effettuato, nel programma Basic, dalla linea 3000 alla linea 3290.

SalvaDisegno

Predispone il salvataggio del disegno su disco, richiama **SelectType** per scegliere il tipo di figura da salvare e , dopo aver richiesto il nome, richiama la procedura **Salva** che, a seconda della scelta effettuata, salva linee, rettangoli o cerchi.

Salva

Salva il disegno su disco. Il parametro "tipo" indica il tipo di figura da salvare e si comporta come in Carica.

Name è il nome del file da salvare (passato dalla procedura SalvaDisegno), al quale sarà aggiunta la giusta estensione a seconda di "tipo".

L'array da salvare verrà quindi, tramite le procedure Lin2Temp, Ret2Temp e Cer2Temp, travasato nell'array temporaneo Temp.

L'istruzione fopen apre il file che verrà identificato dalla variabile stream. Il ciclo for... da 1 al puntatore, permette di salvare tutti gli elementi della figura prescelta. Le istruzioni itoa convertono gli interi, contenuti in Temp, in stringhe; queste verranno dapprima poste nelle variabili stringa Temp1, Temp2, Temp3 e Temp4 e quindi trascritte su disco tramite l'istruzione fprintf.

Il salvataggio del disegno era effettuato, nel programma Basic, dalla linea 2500 alla linea 2790.

InserimComposizione

Permette di inserire, da tastiera, la composizione del disegno. Anche in questo caso viene richiamata la procedura SelectType, che permette la scelta tra Linee, Rettangoli o Cerchi. In seguito verrà presentata una maschera che chiederà i parametri. Tramite un ciclo **Do... While**, che termina con la digitaziuone di **- 1**, i parametri vengono richiesti da tastiera, **separati da virgole**, e presenti tutti sulla stessa linea - video; alla pressione del tasto Return verranno travasati in Temp.

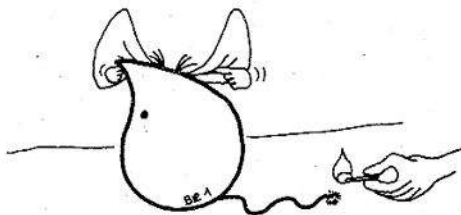
All'introduzione di **- 1**, l'array Temp viene travasato nel giusto array (procedura Temp2Arr) ed il ciclo di inserimento finisce.

La corrispondente procedura, nel listato Basic, è presente dalla linea 1000 alla linea 1130.

Lin2Temp

Travasa, tramite un ciclo For, i contenuti dell'array Linee nell'array Temp ed assegna, al puntatore **Pt**, il valore di **Pl.**

Nel programma Basic questa procedura va dalla linea 9000 alla linea 9060.

Ret2Temp

Travasa, tramite un ciclo For, i contenuti dell'array Rettangoli nell'array Temp ed assegna al puntatore Pt il valore di Pr.

Nel programma Basic, questa procedura va dalla linea 9100 alla linea 9160.

Cer2Temp

Travasa, tramite un ciclo For, i contenuti dell'array Cerchi nell'array Temp ed assegna al puntatore Pt il valore di Pc.

Nel programma Basic, questa procedura va dalla linea 9200 alla linea 9260.

Temp2Arr

Travasa l'array Temp nell'array Linee, Rettangoli o Cerchi a seconda del valore di Temp (=1 Linee; =2 rettangoli; = 3 cerchi). Il puntatore corrispondente (Pl, Pr, Pc) viene assegnato con il valore di Pt. Il travaso viene effettuato tramite un ciclo For, mentre alcune istruzioni If, presenti al suo interno, decidono che cosa e dove copiare.

Nel programma Basic, questa procedura va dalla linea 9300 alla linea 9360.

CancRighe

Cancella, tramite la scrittura di 80 spazi vuoti, le righe video 22 e 23. L'istruzione **Gotoxy (x, y)** serve per spostare il cursore alla posizione desiderata.

Nel programma Basic, vedere le linee da 8850 a 8880.

Lungo o corto?

E' con una certa perplessità che, accontentando le richieste di alcuni (pochi, in verità) lettori, pubblichiamo il lunghissimo listato che appare in queste pagine.

La decisione è stata presa sia per favorire l'apprendimento del **C** (che cosa c'è di più facile che effettuare confronti con linguaggi più familiari, come il Basic?), sia per dimostrare che i linguaggi evoluti richiedono procedure (e pazienza) di impegno maggiore di quanto richiesto da un vetusto interprete.

Ai lettori, come al solito, il compito di commentare adeguatamente la decisione di pubblicare listati completi ma, inevitabilmente, lunghi da digitare.

Per venire incontro ai più pigri, tuttavia, ricordiamo che il listato sorgente è disponibile sulla nostra BBS.

VisArrTemp

Visualizza sullo schermo i contenuti dell'array temporaneo Temp. Il parametro **"i"** è l'indice dal quale partire per la visualizzazione. Tramite un ciclo While verranno visualizzati (procedura **VisRiga**) gli elementi dell'array Temp, a partire dall'indice **i** sino all'indice **i + 15** (15 righe per volta). Il ciclo finisce anche se si raggiunge la fine dell'array.

Vedere corrispondente procedura nel programma basic dalla linea 8900 alla linea 8940.

VisRiga

Visualizza i 4 parametri contenuti in una riga di array temporaneo. Il parametro **"r"** contiene l'indice della riga da visualizzare. La visualizzazione viene effettuata tramite un ciclo For da 0 a 3 ed un'istruzione Printf. La corrispondente procedura nel programma Basic va dalla linea 8950 alla linea 8980.

ModificaComposizione

Richiama le varie procedure di modifica composizione disegno (ModComp2, ModComp3) e ripete sino all'introduzione della scelta 4 (ritorno al menu principale).

ModComp2

Modifica composizione parte 2: richiama la routine **SelectType** per la scelta della figura da modificare, quindi travasa l'array Linee, Rettangoli o Cerchi (a seconda della scelta) nell'array temporaneo.

ModComp3

Modifica composizione parte 3: richiama **VisArrTemp** per la visualizzazione delle righe dell'array da modificare, quindi richiede in input l'operazione desiderata tra: 1) vedere i prossimi 15 elementi, 2) modifica elemento, 3) cancella elemento, 4) vedi disegno, 5) ritorno.

Il ciclo While controlla il corretto inserimento della scelta, quindi, a seconda della scelta effettuata, viene richiamata la procedura **Disegna**, oppure **ModComp4**, che esegue effettivamente la modifica o la cancellazione di un elemento.

ModComp4

Modifica composizione parte 4: permette effettivamente la modifica o la cancellazione di un elemento dell'array prescelto. Il parametro **Ss** indica l'operazione da effettuare (=2 modifica; =3 cancellazione).

Il parametro "i" è l'indice iniziale delle 15 righe visualizzate su video.

Dapprima la procedura chiede l'indice dell'elemento da modificare, quindi, dopo aver controllato (While) che l'indice rientra nei limiti richiesti, richiama la procedura **Lampeggio**, che fa lampeggiare l'oggetto prescelto.

Se l'operazione richiesta è una cancellazione, viene chiesta (scanf) una conferma, che dovrà valere 1 in caso affermativo. Se l'elemento da cancellare era l'ultimo, viene semplicemente decrementato il puntatore dell'array temporaneo Pt, quindi Temp viene travasato nel giusto array (routine Temp2Arr) e la procedura termina. Se non è l'ultimo elemento, tutti gli elementi dell'array temporaneo, posizionati successivamente, vengono spostati, tramite un ciclo For, di una posizione indietro, quindi viene decrementato il puntatore Pt e richiamata la routine Temp2Arr che travasa Temp nel giusto array. Se l'operazione richiesta è una modifica, vengono chiesti i 4 nuovi dati da sostituire, quindi immessi in Temp e poi, sempre tramite la procedura Temp2Arr, Temp viene travasato nel giusto array.

Tutta la parte di modifica composizione disegno, si può trovare nel programma Basic dalla linea 1500 alla linea 1920.

Disegna

Questa routine predispone lo schermo in grafica e quindi, tramite tre cicli For, disegna tutte le linee, tutti i rettangoli e tutti i cerchi. L'istruzione **Detectgraph** determina automaticamente il tipo di **scheda grafica** presente sul computer, e restituisce sia il parametro da utilizzare per "entrare" in grafica, che il nome del driver per la grafica presente su disco.

L'istruzione **Initgraph** mette lo schermo in grafica, utilizzando i parametri preparati da detectgraph. L'ultimo parametro **"c:\\tc"** è un percorso, ed indica il drive e la directory dove si trova il driver grafico (si tratta dei files con estensione **.BGI**, forniti con il pacchetto **TurboC**). Se questo parametro è nullo ("") il compilatore cercherà il driver grafico nella directory corrente. Il driver per la scheda CGA si chiama **CGA.BGI**, quello per la EGA e per la VGA **EGAVGA.BGI**; il driver per la Hercules **HERC.BGI**.

I 3 cicli For successivi richiamano le routines **DrawLine**, **DrawRett** e **DrawCircle** per disegnare, rispettivamente, linee, rettangoli e cerchi. La routine *Disegna* si può trovare nel programma Basic dalla linea 10000 alla linea 11000.

DrawLine, DrawRett, DrawCircle

Disegna o cancella una linea (un rettangolo o un cerchio). Il parametro **k** è l'indice dell'array Linee (Rettangoli, Cerchi) ove sono contenuti i parametri della figura da disegnare. Il parametro **C** indica il colore, se vale **1** la figura viene disegnata mentre se vale **0** viene cancellata.

Le corrispondenti routines Basic si trovano, rispettivamente, dalla linea 30000 alla linea 30030; dalla 31000 alla 31030 e dalla 32000 alla 32030. Per quanto riguarda il cerchio, questo viene disegnato tramite l'istruzione **Ellipse**. I numeri **0** e **360** indicano l'angolo iniziale e l'angolo finale del cerchio da disegnare. Il raggio **Y** viene calcolato dall'aspetto tramite la formula...

Ry = Ra / Asp

...in cui **Ra** è il raggio. Sulla scheda CGA, l'aspetto deve valere circa 2 per riprodurre cerchi perfetti.

Lampeggio

Fa lampeggiare la figura desiderata. Il parametro **Tipo** indica la figura (=1 linea, =2 rettangolo, =3 cerchio). Il parametro **k** è l'indice dell'array dove sono contenuti i dati della figura da far lampeggiare. Il lampeggio è ottenuto tramite un ciclo While.

Dapprima si richiama la procedura DrawLine, DrawRett, o DrawCircle (a seconda del valore di tipo) con colore 1, che quindi disegna la figura, poi, dopo un certo tempo di attesa (Ciclo For da 1 a 70000) si richiama la stessa procedura con Colore 0, che cancella la figura stessa.

Il ciclo si ripete fino a quando viene premuto un tasto (istruzione **kbhit**). Alla fine del lampeggio, **closegraph** rimette lo schermo in modo testo.

Nel programmma Basic, la routine di lampeggio è posta dalla riga 35000 alla riga 35110.

Testo

Aspetta la pressione di un tasto (**Getch**) e rimette lo schermo in modo testo (closegraph). Vedere linee da 9600 a 9630 nel programma Basic.

CancellaDisegno

Cancella il disegno presente in memoria. Viene dapprima richiamata la procedura SelectType per decidere che cosa cancellare.

Poi, a seconda della scelta effettuata, vengono azzerati i puntatori **Pl** (linee), **Pr** (rettangoli) o **Pc** (cerchi), ed infine viene azzerato il puntatore dell'array temporaneo **Pt**.

Nel programma Basic, la routine di Cancella si può trovare dalla linea 2000 alla linea 2080.

Main

L'ultima parte del programma C, paradossalmente, è... il programma principale!

Questa è effettivamente la parte eseguita all'atto del richiamo del programma, e si occupa di formare la maschera iniziale su video; richiedere l'opzione desiderata; richiamare una tra le routines CaricaDisegno, SalvaDisegno, ModificaComposizione, InserimComposizione, Disegna, CancellaDisegno; terminare il programma e tornare in DOS.

Nel programma Basic, il main è posto a partire dalla linea 10 alla linea 500.

```
/*
   MICRO-CAD      "C" Version
   per IBM e Compatibili con Turbo-C 2.0 Borland
   by Mariani Giancarlo 1990
*/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <conio.h>
#include <graphics.h>

const char *T[3]={" Linee"," Rettangoli"," Cerchi"};
/* sp80 = 80 spazi */
const char sp80[81]="
";

unsigned int Linee[51][4];       /* Linee (X1,Y1,X2,Y2)                     */
unsigned int Rettangoli[51][4]; /* Rettangoli (X1,Y1,X2,Y2)                */
unsigned int Cerchi[51][4];      /* Cerchi (X,Y,R,Aspect)                   */
unsigned int Temp[51][4];        /* Array temporaneo x modifiche,ecc. */

unsigned int Pl;                 /* Puntatore array linee                   */
unsigned int Pr;                 /* Puntatore array rettangoli              */
unsigned int Pc;                 /* Puntatore array cerchi                  */
unsigned int Pt;                 /* Puntatore array temporaneo              */

char ch;
int Scelta;                      /* Variabile per scelta opzione menu */
int Opz;                         /* Variabile per sub. SelectType     */
char Stringa[80];                /* Stringa letta dal file            */

/* Prototipi di procedure e funzioni */

void CaricaDisegno(void);          /* Richiama il caricamento da disco    */
void Carica(int tipo,char *name);/* Carica da disco                      */
/* SelectType - Permette all'operatore di scegliere tra linee, rettangoli e cerchi */
void SelectType(int inizio,int *Opz,char *Stringa);
void InserimComposizione(void);  /* Inserisce composizione disegno       */
void Lin2Temp(void);  /* Trasferisce l'array Linee[] nell'array Temp[] */
void Ret2Temp(void);  /* Stessa cosa per l'array Rettangoli[]          */
void Cor2Temp(void);             /* Stessa cosa per l'array Cerchi[]     */
void Temp2Arr(int type); /* Trasferisce l'array Temp in uno dei 3 array sopra.*/
void VisArrTemp(int i);          /* Visualizza l'array Temp[]            */
void VisRiga(int r);  /* Visualizza una riga dell'array Temp[]         */
void CancRighe(void); /* Cancella le righe 22 e 23 del video           */
void ModificaComposizione(void); /* Modifica composizione disegno        */
void ModComp2(void);             /*   "       parte 2                    */
void ModComp3(void);             /*   "       parte 3                    */
void ModComp4(int S,int i);      /*   "       parte 4                    */
void Disegna(void);    /* Disegna in grafica le linee/cerchi/rettangoli */
void Lampeggio(int k,int tipo);  /* Fa lampeggiare una figura            */
void DrawLine(int k,int C);      /* Disegna una linea                    */
void DrawRett(int k,int C);      /* Disegna un rettangolo                */
void DrawCircle(int k,int C);    /* Disegna un cerchio                   */
void Testo(void);                /* Mette il video in modo testo         */
```

```
void CancellaDisegno(void);       /* Cancella il disegno inserito          */
void SalvaDisegno(void);          /* Richiama il salvataggio su disco      */
void Salva(int tipo,char *Name);  /* Salva il disegno su disco             */

/* Scelta tra tutti,linee,cerchi o rettangoli                */
/* inizio : se=0 e' possibile scegliere tutto il disegno     */
/* Opz : riporta in uscita la scelta effettuata              */
/* Stringa : titolo da scrivere in alto allo schermo         */
void SelectType(int inizio,int *Opz,char *Stringa)
{
   *Opz = 0;
   do
   {
      clrscr(); printf("%s\n",Stringa);         /* Forma la maschera video */
      if (inizio == 0) printf("0 : Tutto\n");
      printf("1 : Linee\n");
      printf("2 : Rettangoli\n");
      printf("3 : Cerchi\n");
      printf("4 : Ritorno al menu principale\n\n");
      printf("Scegli : ");
      scanf("%d",Opz);        /* Aspetta in input l'opzione scelta          */
   }
   while (*Opz<inizio || *Opz>4);    /* Se non e' permessa, ripete         */
   printf("\n");
}

/* Richiama il caricamento disegno da disco */
void CaricaDisegno()
{
   char Nome[20];       /* Nome del file da caricare */

   do
   {
      do
      {
         SelectType(0,&Opz,"CARICAMENTO DISEGNO\n");
         if (Opz==4) break;
         printf("\nNome disegno (* ritorna) : ");
         scanf("%s",Nome);    /* Input nome del disegno */
      }
      while (Nome[0]=='*' || Nome[0]==0);
      printf("\n\nCaricamento in corso...\n");
      if (Opz==0 || Opz==1) Carica(1,Nome); /* Richiama il caricamento appropriato */
      if (Opz==0 || Opz==2) Carica(2,Nome);
      if (Opz==0 || Opz==3) Carica(3,Nome);
   }
   while (Opz != 4);
}

/* Carica il disegno da disco               */
/* Se tipo=1 carica le linee                */
/* Se tipo=2 carica i rettangoli            */
/* Se tipo=3 carica i cerchi                */
/* Name = nome del disegno da caricare      */
void Carica(int tipo,char *Name)
{
   char NomeTemp[20];                 /* Copia temporanea del nome          */
   char X1s[10],X2s[10];              /* Variabili per caric. dati da file */
   char X3s[10],X4s[10];
   FILE *stream;

   strcpy(NomeTemp,Name);     /* Aggiunge l'estensione .LIN, .RET o .CER  */
   if (tipo==1) { strcat(NomeTemp,".LIN"); Pl=0; }
   if (tipo==2) { strcat(NomeTemp,".RET"); Pr=0; }
   if (tipo==3) { strcat(NomeTemp,".CER"); Pc=0; }

   Pt=0;
   stream = fopen(NomeTemp,"rt");  /* Apre il file '....LIN, RET o CER' */
   if (stream != NULL)             /* Se il file esiste.....             */
   {
      fgets(Stringa,90,stream);      /* Carica la prima riga                */
      printf("%s\n",Stringa);        /* e la scrive su video                */
      while (fscanf(stream,"%s",X1s) >=1 )
      {
         Pt++;                       /* Carica sino alla fine del file      */
         fscanf(stream,"%s",X2s);  /* e mette nell'array Temp[]             */
         fscanf(stream,"%s",X3s); fscanf(stream,"%s",X4s);
         Temp[Pt][0] = atoi(X1s); Temp[Pt][1] = atoi(X2s);
         Temp[Pt][2] = atoi(X3s); Temp[Pt][3] = atoi(X4s);
      }
      Temp2Arr(tipo);           /* Travasa l'array Temp nel giusto array  */
   }
   fclose(stream);                    /* Chiude il file                    */
}

/* Richiama il salvataggio disegno su disco*/
void SalvaDisegno()
{
   char Nome[20];                     /* Nome del file da salvare          */

   if (Pr==0 && Pl==0 && Pc==0) return; /* Se non c'e' niente da salvare ritorna */
   do
   {
      do
      {
         SelectType(0,&Opz,"SALVATAGGIO DISEGNO\n");
         if (Opz==4) break;
         printf("\nNome disegno (* ritorna) : ");
         scanf("%s",Nome);          /* Input nome del disegno              */
      }
      while (Nome[0]==0 || Nome[0]=='*');
      printf("\nSalvataggio in corso...\n");  /* Richiama il salvataggio appropriato */
      if ((Opz==0 || Opz==1) && Pl>0) Salva(1,Nome);
      if ((Opz==0 || Opz==2) && Pr>0) Salva(2,Nome);
      if ((Opz==0 || Opz==3) && Pc>0) Salva(3,Nome);
   }
   while (Opz != 4);
}

/* Salva il disegno su disco          */
/* Se tipo=1 salva le linee           */
/* Se tipo=2 salva i rettangoli       */
/* Se tipo=3 salva i cerchi           */
```

```c
/* Name - nome del disegno da salvare  */
void Salva(int tipo,char *Name)
{
   char Temp1[15],Temp2[15];         /* Stringhe di supporto              */
   char Temp3[15],Temp4[15];
   char NomeTemp[20];                /* Copia temporanea del nome file    */
   int k;                            /* Variabile per cicli               */
   FILE *stream;

   strcpy(NomeTemp,Name);  /* Aggiunge l'estensione .LIN, .RET o .CER    */
                           /* o travasa il giusto array nell'array Temp  */
   if (tipo==1) { strcat(NomeTemp,".lin"); Lin2Temp(); }
   if (tipo==2) { strcat(NomeTemp,".ret"); Ret2Temp(); }
   if (tipo==3) { strcat(NomeTemp,".cer"); Cer2Temp(); }

   stream = fopen(NomeTemp,"wt");  /* Apre il file                       */
   fprintf(stream,"%s del disegno: %s\n",T(tipo-1),Name); /* 1' riga     */
   for (k=1; k<=Pt; k++)
   {
      itoa(Temp[k][0],Temp1,10);     /* Converte i dati di Temp in stringhe*/
      itoa(Temp[k][1],Temp2,10);
      itoa(Temp[k][2],Temp3,10);
      itoa(Temp[k][3],Temp4,10);
      fprintf(stream," %s, %s, %s, %s\n",Temp1,Temp2,Temp3,Temp4);  /* Scrive una riga sul
file */
   }
   fclose (stream);                  /* Chiude il file                    */
}

/* Inserimento composizione disegno                */
/* Inserisce da tastiera la composizione del disegno */
void InserimComposizione()
{
   int X1,X2,X3,X4;                  /* Variabili di supporto             */

   X1=0; X2=0; X3=0; X4=0;
   do
   {
      SelectType(1,&Opz,"INSERIMENTO COMPOSIZIONE DISEGNO\n");
      if (Opz==4) break;
      clrscr(); Pt=0;
      if (Opz==1) Lin2Temp();        /* Travasa il giusto attay in Temp   */
      if (Opz==2) Ret2Temp();
      if (Opz==3) Cer2Temp();
      printf ("%d %s presenti\n\n",Pt,T(Opz-1));
      printf("Per ogni INPUT inserisci : ");
      if (Opz==1 || Opz==2) printf("X1,Y2,X2,Y2\n");
      if (Opz==3) printf("X,Y,Raggio,Aspetto\n");
      printf("Inserisci '-1' per finire\n\n");

      do                             /* Loop di input da tastiera         */
      {
         printf("Numero %d : ",Pt+1);
         scanf("%d,%d,%d,%d",&X1,&X2,&X3,&X4); /* Input di X1,X2,X3,X4    */
/* se si inserisce -1, Temp viene travasato nel giusto array e il ciclo finisce */
         if (X1==-1) { Temp2Arr(Opz); break; }
         Pt++;
         Temp[Pt][0]=X1; Temp[Pt][1]=X2; /* X1,X2,X3,X4 vanno in Temp     */
         Temp[Pt][2]=X3; Temp[Pt][3]=X4;
      }
      while (X1 != -1);
   }
   while (Opz != 4);
}

/* Travasa l'array Linee[][] nell'array Temp[][] */
void Lin2Temp()
{
   int k;

   Pt=Pl;
   if (Pt==0) return;
   for (k=1; k<=Pt; k++)
   {
      Temp[k][0] = Linee[k][0]; Temp[k][1] = Linee[k][1];
      Temp[k][2] = Linee[k][2]; Temp[k][3] = Linee[k][3];
   }
}

/* Travasa l'array Rettangoli[][] nell'array Temp[][] */
void Ret2Temp()
{
   int k;

   Pt=Pr;
   if (Pt==0) return;
   for (k=1; k<=Pt; k++)
   {
      Temp[k][0] = Rettangoli[k][0]; Temp[k][1] = Rettangoli[k][1];
      Temp[k][2] = Rettangoli[k][2]; Temp[k][3] = Rettangoli[k][3];
   }
}

/* Travasa l'array Cerchi[][] nell'array Temp[][] */
void Cer2Temp()
{
   int k;

   Pt=Pc;
   if (Pt==0) return;
   for (k=1; k<=Pt; k++)
   {
      Temp[k][0] = Cerchi[k][0]; Temp[k][1] = Cerchi[k][1];
      Temp[k][2] = Cerchi[k][2]; Temp[k][3] = Cerchi[k][3];
   }
}

/* Travasa l'array Temp[][] nel giusto array   */
/* Se type=1 travasa nell'array Linee[][]      */
/* Se type=2 travasa nell'array Rettangoli[][] */
/* Se type=3 travasa nell'array Cerchi[][]     */
```

```
void Temp2Arr(int type)
{
   int k;                        /* Variabile per ciclo              */

   if (Pt==0) return;            /* Se non c'e' niente da copiare ritorna
*/
   if (type==1) Pl=Pt;           /* Assegna i puntatori degli array
*/
   if (type==2) Pr=Pt;
   if (type==3) Pc=Pt;
   for (k=1; k<=Pt; k++)         /* Ciclo di copia dell'array
*/
     {
       if (type==1) { Linee[k][0]=Temp[k][0]; Linee[k][1]=Temp[k][1]; }
       if (type==1) { Linee[k][2]=Temp[k][2]; Linee[k][3]=Temp[k][3]; }
       if (type==2) { Rettangoli[k][0]=Temp[k][0]; Rettangoli[k][1]=Temp[k][1];
}
       if (type==2) { Rettangoli[k][2]=Temp[k][2]; Rettangoli[k][3]=Temp[k][3];
}
       if (type==3) { Cerchi[k][0]=Temp[k][0]; Cerchi[k][1]=Temp[k][1]; }
       if (type==3) { Cerchi[k][2]=Temp[k][2]; Cerchi [k][3]=Temp[k][3]; }
     }
}

/* cancella le righe 22 e 23 del video */
void CancRighe()
{
   gotoxy(1,22); printf("%s",sp80);/* Scrive 80 spazi vuoti
*/
   gotoxy(1,23); printf("%s",sp80);
}

/* Visualizza l'array Temp[][] da i a i+15            */
/* i = indice dal quale partire con la visualizzazione */
void VisArrTemp(int i)
{
   int r;                        /* Variabile per ciclo
*/

   clrscr();                     /* Scrive intestazione
*/
   if (Opz==1 || Opz==2) printf("             X1    Y1    X2    Y2\n");
   if (Opz==3) printf("             X     Y     R     Asp\n\n");
   r=i;                          /* Si comincia da i
*/
   do
     {
       if (r>Pt) break;          /* Se fine array, il ciclo finisce      */
       printf ("Numero %3d : ",r);
       VisRiga(r);               /* Visualizza la riga 'r' dell'array    */
       r++;                      /* Incrementa r                         */
     }
   while (r<i+15);               /* il ciclo finisce dopo 15 righe
*/
}

/* Visualizza una riga di Temp[][]       */
/* r = indice della riga da visualizzare */
void VisRiga(int r)
{
   int k;                        /* Variabile per ciclo               */

   for (k=0; k<=3; k++) printf(" %3d  ",Temp[r][k]);
   printf("\n");
}

/* Richiama le routines di Vedi-Modifica composizione disegno */
void ModificaComposizione()
{
   /* Se non e' stato inserito niente, ritorna */
   if (Pl==0 && Pc==0 && Pr==0) return;
   do
     {
       ModComp2(); if (Opz==4) break;
       ModComp3();
     }
   while (Opz != 4);
}

/* Vedi-modifica parte 2: prepara la maschera e travasa il giusto */
/*                        array in Temp[][].                      */
void ModComp2()
{
   int k;

   do
     {
       SelectType(1,&Opz,"VEDI/MODIFICA COMPOSIZIONE DISEGNO\n");
       if (Opz==4) break;
       printf("\n"); Pt=0;
       if (Opz==1) Lin2Temp();        /* Travasa l'array in Temp[][]     */
       if (Opz==2) Ret2Temp();
       if (Opz==3) Cer2Temp();
       printf ("%d %s presenti\n\n",Pt,T[Opz-1]);
       for (k=1; k<10000; k++);
     }
   while (Pt<=0);
}

/* Vedi-modifica parte 3: Visualizza le righe dell'array e permette di  */
/*                        Scegliere se modificarle, cancellarle, ecc.   */
/* Richiama la visualizzazione in grafica del disegno e il lampeggio di */
/* un oggetto                                                           */

void ModComp3()
{
   int i,Sz;                       /* Variabili di supporto             */

   i=1;                            /* Indice per la visualizzazione     */
   do
     {
       if (i>Pt) i=Pt-15;          /* Controlli sui limiti dell'indice  */
```

```
    if (i<=0) i=1;
    VisArrTemp(i);                  /* Visualizza 15 righe da i a i+15     */
    do
    {
      CancRighe();
      /* Maschera per l'introduzione della scelta */
      gotoxy(1,22); printf("1 = prossimi 15, 2 = mod., 3 = canc.\n");
      gotoxy(1,23); printf("4 = vedi dis., 5 = ritorno ");
      scanf("%d",&Ss);              /* Input scelta                        */
    }
    while (Ss<1 || Ss>5);           /* Se scelta errata, ripete            */
    if (Ss==4) { Disegna(); Testo(); }  /* Disegna in grafica              */
    if (Ss==1) i=i+15;              /* Prossimi 15 elementi                */
    if (Ss==2 || Ss==3) ModComp4(Ss,i); /* Richiama modifica parte 4       */
  }
  while (Ss != 5);
}

/* Vedi-modifica parte 4: Permette di scegliere un oggetto da modificare o */
/* cancellare, lo fa lampeggiare sullo schermo grafico e quindi lo modifica*/
/* o cancella.                                                             */
/* Ss = 2 : richiesta di modifica                                          */
/* Ss = 3 : richiesta di cancellazio...                                    */
/* i = indice dell'inizio delle righe dell'array visualizzate              */
void ModComp4(int Ss,int i)
{
  int N,S,k,X1,X2,X3,X4;            /* Variabili di supporto               */

  do
  {
    CancRighe();   /* Richiesta indice oggetto da cancellare o modificare  */
    gotoxy(1,22); printf("Numero : "); scanf("%d",&N);
    if (N==0) break;
  }
  while (N<1 || N>Pt);
  if (N==0) return;                 /* Se si inserisce 0, ritorna          */
  Lampeggio(N,Opz);                 /* Lampeggio oggetto prescelto         */
  if (Ss==3)                        /* Cancellazione oggetto               */
  {
    VisArrTemp(i); CancRighe();     /* Chiede una conferma                 */
    gotoxy(1,22); printf("1 = cancella, 0 = non cancella : ");
    scanf("%d",&S); if (S!=1) return;
    if (N>Pt) { Pt--; Temp2Arr(Opz); return; }
    for (k=N+1; k<=Pt; k++)         /* Aggiusta array dopo cancellazione   */
    {
      Temp[k-1][0]=Temp[k][0]; Temp[k-1][1]=Temp[k][1];
      Temp[k-1][2]=Temp[k][2]; Temp[k-1][3]=Temp[k][3];
    }
    Pt--; Temp2Arr(Opz); return;
  }
  else                              /* Modifica dell'oggetto               */
  {
    VisArrTemp(i); CancRighe();
    gotoxy(1,22);
    if (Opz==1 || Opz==2) printf("X1,Y1,X2,Y2 ");
    if (Opz==3) printf("X,Y,Raggio,Asp. ");
    printf("n. %d (-1=fine)\n",N);/* Richiede in input i nuovi dati        */
    gotoxy(1,23);scanf("%d,%d,%d,%d",&X1,&X2,&X3,&X4);
    if (X1==-1) return;             /* Se si inserisce -1, non modifica    */
    Temp[N][0]=X1; Temp[N][1]=X2; /* Immette i nuovi dati nell'array Temp  */
    Temp[N][2]=X3; Temp[N][3]=X4;
    Temp2Arr(Opz);                  /* Travasa Temp[][] nel giusto array   */
  }
}

/* Disegna sullo schermo grafico                     */
/* I dati per il disegno vengono presi dagli arrays : */
/* Linee[][], Corchi[][], Rettangoli[][]             */
/* Puntatori degli array : Pl,Pr,Pt                  */
void Disegna()
{
  int g_driver, g_mode,k;           /* Variabili per tipo grafica          */

  detectgraph(&g_driver, &g_mode);/* Determina la scheda grafica presente  */
  /* Mette lo schermo in grafica. NB : "c:\tc" = directory dove sono       */
  /* presenti i files del Turbo-C "*.BGI"                                  */
  initgraph(&g_driver, &g_mode, "c:\\tc");
  cleardevice();                    /* Cancella lo schermo grafico         */

  if (Pl>0) for (k=1; k<=Pl; k++) DrawLine(k,1);  /* Disegna le linee      */
  if (Pr>0) for (k=1; k<=Pr; k++) DrawRett(k,1);  /* Disegna i rettangoli  */
  if (Pc>0) for (k=1; k<=Pc; k++) DrawCircle(k,1);/* Disegna i cerchi      */
}

/* Disegna o cancella una linea             */
/* k=indice della linea nell'array Linee[][] */
/* C = 0 : cancella                         */
/* C = 1 : disegna                          */
void DrawLine(int k,int C)
{
  int X1,X2,Y1,Y2;                  /* Variabili di supporto               */

  X1 = Linee[k][0]; Y1 = Linee[k][1];
  X2 = Linee[k][2]; Y2 = Linee[k][3];
  setcolor (C);                     /* Setta il colore                     */
  line (X1, Y1,X2, Y2);             /* Traccia la linea                    */
}

/* Disegna o cancella un rettangolo                   */
/* k=indice del rettangolo nell'array Rettangoli[][] */
/* C = 0 : cancella                                  */
/* C = 1 : disegna                                   */
void DrawRett(int k,int C)
{
  int X1,X2,Y1,Y2;                  /* Variabili di supporto               */

  X1 = Rettangoli[k][0]; Y1 = Rettangoli[k][1];
  X2 = Rettangoli[k][2]; Y2 = Rettangoli[k][3];
  setcolor (C);                     /* Setta il colore                     */
  rectangle (X1,Y1,X2,Y2);          /* Disegna il rettangolo               */
}
```

```
/* Disegna o cancella un cerchio                        */
/* k=indice del cerchio nell'array Cerchi[][] */
/* C = 0 : cancella                                     */
/* C = 1 : disegna                                      */
void DrawCircle(int k,int C)
{
   int X,Y;                          /* Variabili di supporto      */
   float Yradius;                    /* Raggio Y                   */
   float Ra,Asp;                     /* Raggio e aspetto           */

   X = Cerchi[k][0]; Y = Cerchi[k][1];
   Ra = Cerchi[k][2]; Asp = Cerchi[k][3];
   Yradius = Ra/Asp;                 /* Calcola il raggio Y        */
   setcolor (C);                     /* Setta il colore            */
   ellipse (X,Y,0,360,Ra,Yradius); /* Disegna il cerchio          */
}

/* fa lampeggiare l'elemento che si vuole modificare o cancellare */
/* k = indice dell'elemento da far lampeggiare                    */
/* tipo = 1 : elemento = Linea                                    */
/* tipo = 2 : elemento = Rettangolo                               */
/* tipo = 3 : elemento = Cerchio                                  */
void Lampeggio(int k,int tipo)
{
   int C;                            /* Colore                     */
   long i;                           /* Variabile per conteggio    */

   Disegna(); C = 0;                 /* Colore 0 = cancella        */
   do
   {
      if (tipo==1) DrawLine(k,C);   /* Alternativamente cancella/disegna */
      if (tipo==2) DrawRett(k,C);   /* l'elemento prescelto       */
      if (tipo==3) DrawCircle(k,C);
      for (i=1; i<=70000; i++);     /* Tempo di attesa (dipende dal computer)*/
      C = 1-C;                      /* Inverte il colore          */
   }
   while (kbhit() == 0);             /* Se non si preme un tasto, continua */
   getch();                          /* Svuota il buffer di tastiera */
   closegraph();                     /* Rimette lo schermo in testo */
}

/* Aspetta la pressione di un tasto e mette in testo */
void Testo()
{
   getch();                          /* Aspetta la pressione di un tasto */
   closegraph();
}

/* Cancellazione disegno dalla memoria */
void CancellaDisegno()
{
   /* Se non e' presente un disegno, ritorna */
   if (Pr==0 && Pl==0 && Pc==0) return;
   SelectType(0,&Opz,"CANCELLAZIONE DISEGNO\n");
   if (Opz==4) return;
   if (Opz==0 || Opz==1) Pl=0;      /* Azzera il giusto puntatore del tipo */
   if (Opz==3 || Opz==2) Pr=0;      /* di figura da cancellare    */
   if (Opz==0 || Opz==3) Pc=0;
   Pt=0;                             /* Azzera il puntatore di Temp[][] */
}

/* ----------- Main Program ----------- */
void main()
{
   Pl=0; Pr=0; Pc=0; Pt=0;           /* Azzeramento puntatori      */

   do
   {
      do
      {
         clrscr();                   /* Cancella lo schermo        */
         /* Maschera iniziale        */
         printf("MICRO-CAD for IBM & Borland's Turbo-C 2.0\n");
         printf("by Gianky'90\n\n");
         printf("1 : Carica Disegno\n");
         printf("2 : Salva Disegno\n");
         printf("3 : Vedi/modifica composizione disegno\n");
         printf("4 : Inserimento composizione disegno\n");
         printf("5 : Vedi grafica disegno\n");
         printf("6 : Cancellazione disegno\n");
         printf("7 : Fine CAD\n\n");
         printf("Scegli : ");        /* Aspetta in input la scelta */
         scanf("%d",&Scelta);
      }
      while ((Scelta < 1) || (Scelta > 7));  /* Ripete se scelta errata */

      if (Scelta == 1) CaricaDisegno();     /* Richiama la giusta routine */
      if (Scelta == 2) SalvaDisegno();      /* a seconda della scelta     */
      if (Scelta == 3) ModificaComposizione();
      if (Scelta == 4) InserimComposizione();
      if (Scelta == 5) { Disegna(); Testo(); }
      if (Scelta == 6) CancellaDisegno();
   }
   while (Scelta != 7);
   clrscr();                         /* Fine programma             */
   printf("Fine CAD...\n");
}
```

INDOVINA, INDOVINELLO

Chi porta a casa un computer per la prima volta vorrebbe stupire (e, magari, divertire) i propri familiari. Ecco un programmino di sicuro "effetto"...

Sul n. 7 di Commodore Computer Club (del dicembre **1983**, quasi nella preistoria...) fu presentato un listato per il Pet 2000, davvero interessante e, soprattutto, facilissimo da digitare.

In queste pagine lo ripubblichiamo in due versioni.

La prima è valida per l'**Amiga** ed i computer **Ms-Dos** (dotati di linguaggio **Quick Basic** Microsoft); la seconda, invece, per il **C/64** (dotato di emulatore Gw-Basic) e per computer Ms-Dos su cui gira il più vetusto (ma molto difuso) linguaggio interprete **Gw-Basic**, noto in moltissime versioni.

I due listati

Per la versione Gw-Basic c'è da dire, soltanto, che alcune note sul listato sono riportate, in evidenza, nelle **Rem**.

Chi possiede il C/64 (dotato di **emulatore Gw-Basic**) oppure un computer Ms-Dos, non deve fare altro che "caricare" il linguaggio interprete e digitare le righe, una per una, non dimenticando di premere il tasto **Return** al termine di ogni riga Basic.

Anche chi possiede un Amiga oppure un Ms-Dos dotato di Quick Basic (una qualsiasi versione dei linguaggi interpreti sarà valida per il corretto funzionamento del programma) deve dapprima caricare il linguaggio interprete; quindi digitare le righe che, nel listato, sono indicate come indispensabili per il funzionamento del programma.

In particolare, gli utenti Amiga **non** devono digitare quanto, appunto, indicato valido per il solo Quick Basic.

Che cosa fa il programma

Il listato consente di generare un archivio di **"oggetti"** (per esempio, nomi di vari animali) ognuno dei quali è caratterizzato da uno (o più) **"attributi"** specifici. Successivamente il computer ci invita a pensare ad uno di tali nomi e pone varie domande (esempio: miagola? ha le pinne? vive nella steppa?, è un animale domestico? eccetera).

In base alle risposte che noi forniamo (semplicemente **S** per una risposta affermativa e **N** per una negativa) giunge alle conclusioni fornendo il nome dell'oggetto che, secondo "lui", abbiamo in mente.

Il risultato delle operazioni può sembrare banale, ma vi assicuriamo che, se memorizzate un numero elevato di nomi (potete registrarne fino a **300**, ed anche più!) ad ognuno dei quali attribuite caratteristiche "insolite", il divertimento è assicurato.

```
REM Indovina, indovinello
REM Mini gioco per principianti (e non)
REM versione per AMIGABASIC e linguaggio QUICK BASIC
```

DECLARE SUB LeggeDisco ()	: REM solo per **Quick Basic**
DECLARE SUB tasto ()	: REM solo per **Quick Basic**
DECLARE SUB HoIndovinato ()	: REM solo per **Quick Basic**

```
DIM SHARED a$, i, a$(300), a%(150): X$ = "(s/n)"
CLS : PRINT "1- Carico da disco": PRINT "2- Inizio un nuovo iter ";
CALL tasto
    IF a$ = "1" THEN
        CALL LeggeDisco: j = i + 1:
    END IF
 IF a$ = "2" THEN
    INPUT "Digita il primo nome della serie"; b$
    a$(0) = "@" + b$: i = 1
 END IF
inizia:
i = 0: CLS : PRINT "Pensa a ci che devo indovinare..."
PRINT "... e quando sei pronto premi un tasto ";
CALL tasto
richiedi:
    IF ASC(a$(i)) = 64 THEN
        CALL HoIndovinato
            IF UCASE$(a$) = "S" THEN
                PRINT "Vuoi giocare ancora "; X$; : CALL tasto
                    IF UCASE$(a$) = "S" THEN
                        GOTO inizia:
                        ELSE
                        GOTO Registrazione
                    END IF
            END IF: GOTO NonIndovinato
    END IF
PRINT a$(i); "? "; : CALL tasto
    IF UCASE$(a$) = "S" THEN i = a%(i): GOTO richiedi
i = a%(i) + 1: GOTO richiedi
NonIndovinato:
INPUT "Che cos'era"; b$
PRINT "Che domanda faresti parlando di un "; b$; : INPUT c$
PRINT "Che risposta daresti? "; X$; : INPUT d$
    IF UCASE$(d$) = "S" THEN
        a$(j) = "@" + b$
        a$(j + 1) = a$(i)
    END IF
 IF UCASE$(d$) = "N" THEN
    a$(j + 1) = "@" + b$
    a$(j) = a$(i)
 END IF
a$(i) = c$: a%(i) = j
j = j + 2: a$(j) = "FINE"
PRINT "Vuoi giocare ancora? "; : CALL tasto
    IF UCASE$(a$) = "S" THEN GOTO inizia
Registrazione:
PRINT "Devo registrare? "; : CALL tasto
    IF UCASE$(a$) = "N" THEN END
```

Come gira il programma

Per capire in che modo gira il programma, vi suggeriamo di seguire alla lettera l'esempio che vi proponiamo; in seguito sarete in grado di arrangiarvi da soli creando vari archivi di *domande / risposte* utili per giocare in famiglia.

Non appena digitate **Run**, compare un semplice menu di scelta. Dal momento che è la prima volta che fate girare il programma, rispondete premendo il tasto **2** *(Inizio un nuovo iter)*.

Supponiamo di voler realizzare un archivio di **animali**.

Alla domanda *"Digita il primo nome della serie"*, rispondete con **Cane**. Non appena premete il tasto Return il computer vi chiederà *"Pensa a ciò che devo indovinare e premi un tasto"*.

Se, ad esempio, avete pensato al **gatto**, dopo aver premuto un tasto qualsiasi il computer visualizzerà l'unico nome che ha in memoria *(cane)* e chiederà se ha indovinato.

Naturalmente risponderete **N** e, subito dopo, verrà posta automaticamente la domanda *"Che cos'era?"*. Dopo aver risposto **Gatto**, il computer chiederà ancora: *"Che domanda faresti parlando di un gatto?"*.

Rispondendo con **Miagola,** il computer chiederà, subito dopo, se a questa domanda la risposta è **sì** oppure **no**.

Naturalmente digiterete **S** e allo stesso modo risponderete alla domanda successiva *(Vuoi giocare ancora?)*.

A questo punto il computer ha in memoria due nomi e porrà subito l'unica domanda che è in grado di porre:

Miagola?

Se rispondete **S** apparirà la domanda *"E' un gatto?"*. L'animale al quale corrisponde una domanda affermativa, infatti, è (almeno per ora) solo il gatto.

Se, invece, rispondete **N** alla domanda di prima (Miagola?) il computer tenterà con l'altro nome a disposizione *(E' un cane?)* e se rispondete **N** chiederà la digitazione del nuovo nome, ripetendo il ciclo.

Per farla breve, ecco una procedura che vi proponiamo nel vostro primo esperimento, tenendo presente che in *carattere corsivo* sono indicate le visualizzazioni del programma, in **carattere neretto**, invece, sono indicate le risposte che dovete digitare:

```
PRINT : PRINT "Da quale drive?"
  PRINT "Per Amiga: DF0: oppure DF1:"
  PRINT "Per Ms-Dos: A: oppure B: eccetera"
  INPUT "(non dimenticare i due punti...) "; dv$
  INPUT "Nome del file"; nf$
      i = 0: OPEN dv$ + nf$ FOR OUTPUT AS #1
      WHILE i <= j: PRINT #1, a$(i): PRINT #1, a%(i): i = i + 1
  WEND: CLOSE #1: END

  SUB HoIndovinato STATIC
  PRINT RIGHT$(a$(i), LEN(a$(i)) - 1)
  PRINT "Ho indovinato? ";
  CALL tasto
  END SUB

  SUB LeggeDisco STATIC
   PRINT : PRINT "Da quale drive?"
   PRINT "Per Amiga: DF0: oppure DF1:"
   PRINT "Per Ms-Dos: A: oppure B: eccetera"
   INPUT "(non dimenticare i due punti...) "; dv$
   INPUT "Nome del file"; nf$
   OPEN dv$ + nf$ FOR INPUT AS #1: i = 0
      WHILE EOF(1) = 0
         IF a$(i) <> "FINE" THEN
            INPUT #1, a$(i): INPUT #1, a%(i)
            i = i + 1
         END IF
      WEND
   CLOSE #1
   END SUB

  SUB tasto STATIC
  a$ = "": WHILE a$ = ""
  a$ = INKEY$
  WEND: PRINT a$
  END SUB
```

Pensa a ciò che devo indovinare...
è un cane?
N
Che cos'era?
Gatto
Che domanda faresti sul gatto
miagola
Come risponderesti (s/n)"
s
Giochi ancora?
s
Pensa a ciò che devo indovinare...
Miagola?
n
Che cos'era?
gallina
Che domanda faresti sulla gallina
E' un animale domestico
Come risponderesti (s/n)
s
...eccetera

Insomma, ad ogni nome nuovo che digitate, il computer memorizzerà la domanda ad esso relativa e la risposta che le compete per individuarlo in seguito. Volendo, è possibile attribuire più caratteristiche ad uno stesso nome digitando più di una domanda (e relative risposte).

Naturamente il programma non si limita a memorizzare nomi di animali, ma anche (e soprattutto) di **amici** e familiari.

Se, infatti, al nome dei vostri amici attribuite caratteristiche... divertenti, vi assicuriamo che sarà un vero spasso costringere il computer ad indovinare il nome, dal momento che porrà, una dopo l'altra, tutte le domande a sua disposizione. Un esempio simpatico di domande visualizzate può, dopo una serata passata al programma, essere il seguente...

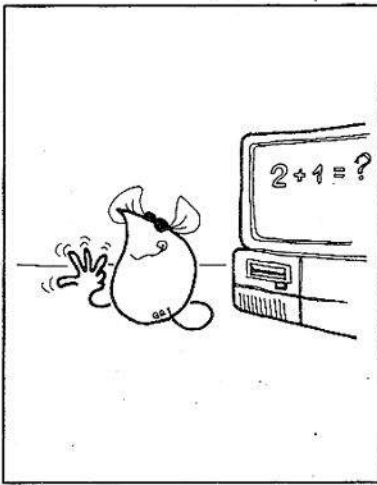
Ha le gambe storte?
E' strabica?
Veste in modo approssimativo?
E' intelligente?
E' un animale domestico?
Ha la cariinga stretta e lunga?
Ha la ruota di scorta?
E' Giovanna?

Non è infatti necessario che il gruppo di nomi sia omogeneo; potete quindi mescolare nomi di **amici**, di **animali**, di **oggetti** e così via.

Dal momento che il programma pone una domanda o un'altra a seconda della risposta che è stata fornita, la situazione diventa divertente quando vengono memorizzate una ventina di domande e relative risposte.

Alla fine, per non perdere il contenuto della memoria, è possibile registrare su disco l'intero archivio. In seguito è possibile richiamarlo e giocare e/o aggiungere altri nomi.

Migliorie

La diversità esistente tra la sintassi di **AmigaBasic** e quella di **Quick Basic** non ha consentito lo sfruttamento adeguato delle risorse di entrambi gli interpreti. In particolare, non si è potuto evitare l'irrazionale ricorso alle istruzioni di tipo **GoTo**, poco efficienti ed eleganti.

Cosigliamo ai lettori più "evoluti" (e pazienti) di apportare le dovute modifiche in modo da rendere ottimale l'uso del programma. In particolare consigliamo di inserire una routine che eviti l'interruzione del programma in caso di errore di digitazione dei dati al momento della registrazione dell'archivio.

Utile sarà anche la presenza di una routine che consenta di ritornare sui propri passi in caso di errata digitazione di una domanda (e/o relativa risposta).

Sfida ai lettori

La procedura descritta, grazie anche alla notevole brevità dei listati, si presta benissimo per esser implementata in altri linguaggi. Ci riferiamo, ovviamente, al **Turbo Pascal** (caso dei computer Ms-Dos) ed al linguaggio **"C"** (disponibile sia per Amiga che per Ms-Dos). I lettori che fossero in grado di realizzare una o l'altra implementazione (purchè **brevissima**) possono inviare il dischetto contenente il programma; nello stesso dischetto deve esser presente una breve **descrizione** del funzionamento del programma, redatta con un qualsiasi **word processor**.

L'autore della versione giudicata migliore (che verrà pubblicata, ovviamente, sulla nostra rivista) riceverà gratis, per un anno, Commodore Computer Club (o altra rivista edita dalla Systems Editoriale).

```
100 REM Indovina, indovinello
110 REM Mini gioco per principianti (e non)
120 REM versione per GW-BASIC (Ms-Dos oppure C/64
+ emulatore Gw-Basic)
130 DIM A$(300), A%(150): X$ = "(s/n)"
140 CLS : PRINT "1- Carico da disco": PRINT "2- Inizio un
nuovo iter ";
150 GOSUB 750
160 IF A$ = "1" THEN GOSUB 650: J = I + 1:GOTO 210
170 INPUT "Digita il primo nome della serie"; B$
180 A$(0) = "@" + B$: J = 1
190 :
200 REM INIZIA:
210 I = 0: PRINT "Pensa a ci che devo indovinare..."
220 PRINT "... e quando sei pronto premi un tasto ";
230 GOSUB 750
240 :
250 REM RICHIEDI
260 IF ASC(A$(I)) <> 64 THEN 320
270 GOSUB 600 :REM (HO INDOVINATO?)
280 IF A$="N" OR A$="n" THEN GOTO 370
290 PRINT "Vuoi giocare ancora "; X$; : GOSUB 750
300 IF A$ = "S" OR A$="s" THEN GOTO 210
310 GOTO 480 :REM (REGISTRAZIONE)
320 PRINT A$(I); "? "; : GOSUB 750
330 IF A$ = "s" OR A$="S" THEN I = A%(I): GOTO 260
340 I = A%(I) + 1: GOTO 260: REM (RICHIEDI)
350 :
360 REM NONINDOVINATO:
370 INPUT "Che cos'era"; B$
380 PRINT "Che domanda faresti parlando di un "; B$; :
INPUT C$
390 PRINT "Che risposta daresti? "; X$; : INPUT D$
400 IF D$ = "S" OR D$="s" THEN A$(J) = "@" + B$:A$(J
+ 1) = A$(I)
410 IF D$ = "N" OR D$="n" THEN A$(J + 1) = "@" +
B$:A$(J) = A$(I)
420 A$(I) = C$: A%(I) = J
430 J = J + 2: A$(J) = "FINE"
440 PRINT "Vuoi giocare ancora? "; : GOSUB 750
450 IF A$="s" OR A$ = "S" THEN GOTO 210
460 :
470 REM REGISTRAZIONE:
480 PRINT "Devo registrare? "; : GOSUB 750
490 IF A$="n" OR A$ = "N" THEN END
500 PRINT : PRINT "Da quale drive?"
510 PRINT "Per Amiga: DF0: oppure DF1:"
520 PRINT "Per Ms-Dos: A: oppure B: eccetera"
530 INPUT "(non dimenticare i due punti...) "; DV$
540 INPUT "Nome del file"; NF$
550 I = 0: OPEN DV$ + NF$ FOR OUTPUT AS #1
560 WHILE I <= J: PRINT #1, A$(I)
565 PRINT #1, A%(I): I = I + 1
570 WEND: CLOSE #1: END
580 :
590 REM SUBROUTINE HOINDOVINATO
600 PRINT RIGHT$(A$(I), LEN(A$(I)) - 1)
610 PRINT "Ho indovinato? ";
620 GOSUB 750:RETURN
630 :
640 REM SUBroutine LEGGEDISCO
650 PRINT : PRINT "Da quale drive?"
660 PRINT "Digita... A: oppure B: eccetera"
670 INPUT "(non dimenticare i due punti...) "; DV$
680 INPUT "Nome del file"; NF$
690 OPEN DV$ + NF$ FOR INPUT AS #1: I = 0
700 WHILE EOF(1) = 0
710 IF A$(I) <> "FINE" THEN INPUT #1, A$(I): INPUT #1,
A%(I):I = I + 1
720 WEND: CLOSE #1: RETURN
730 :
740 REM SUBroutine TASTO
750 A$ = "": WHILE A$ = ""
760 A$ = INKEY$
770 WEND: PRINT A$
780 RETURN
```

Versione Gw-Basic

Le righe che, apparentemente, sembrano andare "a capo" (come la riga 380 e la 410), vanno digitate, in realtà, su un'unica riga vide; bisogna sempre premere il tasto Return, insomma, prima di digitare la successiva riga Basic

di Ugo Spezza

GRAFICI DI FUNZIONI TRIDIMENSIONALI

Un programma matematico che affronta uno dei temi più affascinanti della grafica: la visualizzazione delle funzioni matematiche con tanto di gestione delle linee nascoste

Prima di parlare del programma pubblicato in queste pagine, occorre richiamare alcuni concetti di geometria euclidea, allo scopo di invogliare alla digitazione del listato anche gli studenti delle scuole medie inferiori (a patto che siano dotati di buona volontà...).

E' noto che il concetto di "Dimensione uno" è una **retta**, vale a dire che un punto **P (x)** si può muovere solo lungo la direzione della retta stessa e resta individuato dalla distanza da un punto **O** (di coordinata 0) fissato a piacere e denominato convenzionalmente *Origine*.

Operando nelle due dimensioni, invece, si ha a che fare con un **piano**, in cui ci si potrà muovere nelle due direzioni.

Si usa fissare due assi cartesiani, **X** ed **Y**, perpendicolari fra di loro, che si intersecano in un punto O (di coordinate 0, 0) definito come origine.

Un punto **P (x, y)** verrà quindi individuato dalla *distanza* che lo separa dall'asse **X** e dall'asse **Y**. Fin qui, speriamo, nulla di nuovo.

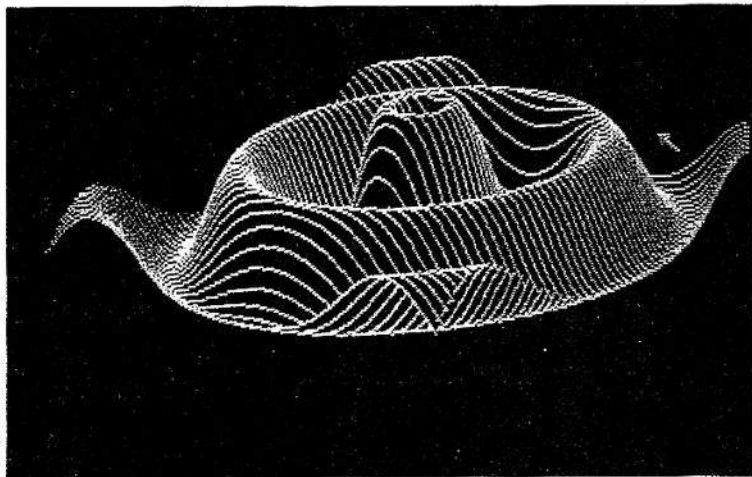
Per quanto riguarda, invece, le **tre** dimensioni, bisogna riferirsi ad un piano su cui sono disegnati i due assi precedenti (X e Y) che si intersecano nel punto origine O; in questo immaginate di conficcare (magari a martellate) un chiodo, il cui prolungamento ideale sarà da noi individuato come **asse Z**.

I tre assi X, Y e Z si in tersecano nell'origine O (0, 0, 0) e sono a 90 gradi tra loro .

E' facile capire che un punto P (di coordinate x, y, z) resta individuato dalle rispettive distanze dai tre assi suddetti.

Per disegnare un oggetto nello spazio tridimensionale bisognerà quindi individuare una legge (o meglio, FUNZIONE) in grado di elaborare il disegno dei punti stessi affinchè venga tracciata la corrispondente immagine.

Se nel piano bidimensionale la funzione era del tipo **y = f (x)**, nello spazio sarà **z = f (x, y)**.

Si potrebbe pensare di disegnare grafici in quattro dimensioni, in cui il tempo assumerebbe il compito di indicare la quarta dimensione; ma in queste pagine ci limiteremo a casi più semplici ed alla portata di tutti, soprattutto se studenti delle scuole superiori.

```
' [FUNZIONI TRIDIMENSIONALI]
' Matematica per Amiga Basic
' by Ugo Spezza

Funzione: DEF FNz (x, y) = SIN (x^2 + y^2)
SCREEN 1, 640, 256, 4, 2
WINDOW 2, SPACE$ (20) + "MODALITA DI VISUALIZZAZIONE DEL GRAFICO: ",, 0, 1
PALETTE 0, 0, 0, 0
PALETTE 2, 0, 1, 0.7: PALETTE 3, 1, 0.4, 0
LOCATE 5, 1: COLOR 3

PRINT " [1] - Disegno dello scheletro."
PRINT : PRINT " [2] - Disegno della superficie."
10 a$ = INKEY$: IF a$ = "" THEN 10
ss = VAL (a$) + 2: IF ss> = 4 THEN ss = 0
LOCATE 10, 1: COLOR 2

INPUT" Densita' delle linee [1]-[2] ";de
ve = de / 10 + 0.1: IF ve>0.2 THEN ve = 0.1
PALETTE 1, 0, 0, 0
infl = 3: ' Inflessione curva
CLS: pi = 3.1416: s = pi / 5: f = infl
sc = COS (5 / 2) : sd = SIN (5 / 2) : cc = COS (pi / 4) : cd = SIN (pi / 4)
IF ss = 0 THEN DIM y (639), q (639) : FOR i = 0 TO 639: y (i) = 255: NEXT i
FOR a = -f TO f STEP ve
FOR b = f TO -f STEP -ve
x = a * s: y = b * s: r = FNz (x, y)
IF r = 0 THEN c = 1.5 ELSE c0 = -2 * SIN (1 * r) / (r / 4 + 1.5) * 2
xs = 320 + (x * cc + y * cd) * 120: ys = 90 + (x * sc + y * sd + c0) * 20
IF b = f THEN xp = xs: yp = ys ELSE GOSUB 20: xp = xs: yp = ys
NEXT b: NEXT a: BEEP

15 b$ = INKEY$: IF b$ = "" THEN 15
SCREEN CLOSE 1: LIST Funzione: END

20 dx = xs-xp: dy = ys-yp: IF ABS (dy) >ABS (dx) THEN 30
dd = dy / dx: t = 1 + ss: IF dx < 0 THEN t = -1-ss
FOR I = xp TO xs STEP t: m = yp + (I-xp) * dd: GOSUB 40: NEXT I: RETURN

30 dd = dx / dy: t = 1 + ss: IF dy < 0 THEN t = -1-ss
FOR m = yp TO ys STEP t: I = xp + (m-yp) * dd: GOSUB 40: NEXT m: RETURN

40 IF ss = 3 THEN scheletro
IF m>q (I) THEN PSET (I, m), 2: q (I) = m: GOTO punti:
IF m < y (I) THEN PSET (I, m), 2: y (I) = m: RETURN

punti:
IF m < y (I) THEN y (I) = m
RETURN

scheletro:
IF I = xp THEN PSET (xp, yp), 3 ELSE LINE - (xs, ys), 3
RETURN
```

Versione AmigaBasic del programma

Il programma

Come si vede nella prima riga del listato (commenti a parte), viene definita, con **DEF FN,** una funzione di due variabili **z = f (x, y)** resposabile del disegno che apparirà sullo schermo di Amiga.

Il programma funziona tracciando rette, parallele al piano **xy**, che verranno poi usate dalla funzione **Z** per sezionare la superficie della curva 3D con una serie di piani paralleli passanti per le rette stesse.

I cicli da elaborare sono quattro: 2 per le coordinate matematiche e 2 per le coordinate video. I due vettori dimensionati serviranno per cancellare le linee retrostanti della superficie (**linee nascoste**); in essi vengono inseriti, di volta in volta, i valori **Y** calcolati durante la determinazione delle coordinate matematiche che verificano l'eventuale "superamento" del valore di Y.

In caso affermativo la linea viene tracciata; in caso contrario (linea posizionata "alle spalle" di quanto già disegnato) la linea non viene evidenziata sullo schermo.

Nel listato è messa in evidenza la variabile **Infl** (= Inflessione) che viene posta eguale a 3. Aumentando o diminuendo tale valore, cambia l'inflessione della curva, ossia l'intervallo a, b (-3, 3) di visualizzazione sul piano XY.

❑❒

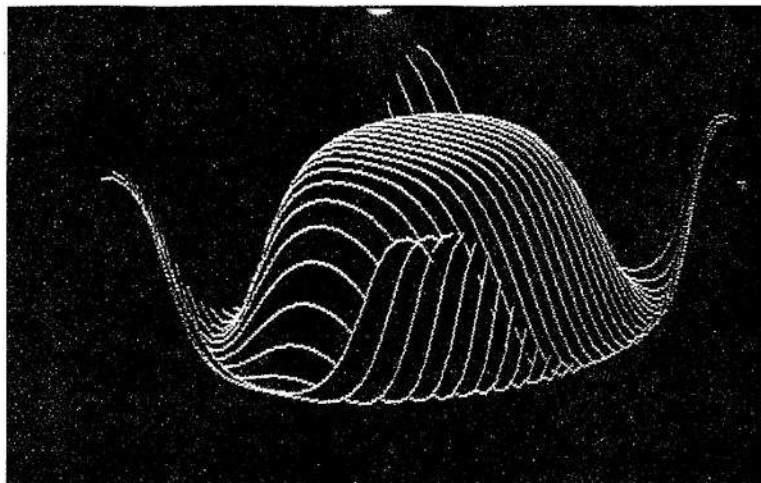
Funzionamento

Dato il Run, viene richiesta la modalità di visualizzazione del grafico 3D e se si desidera vedere lo **Scheletro** della figura o la **Superficie**.

Nel primo caso (scheletro) la curva viene disegnata corredata di tutte le linee retrostanti; selezionando, invece, superficie, viene tracciata la curva 3D vera e propria.

Viene poi richiesta la **Densità** delle linee (**1** oppure **2**). In questo caso occorre fare una scelta tra un'accurata definizione (che, però, richiede un tempo di esecuzione maggiore) ed una minore precisione (compensata da tempi di elaborazione relativamente brevi).

Questi variano tra i 2.5 minuti (per il disegno dello scheletro) a 6 (disegno della superficie); i tempi, inoltre, vanno al-

meno raddoppiati scegliendo densità 2. Prima di compilare i grafici con AC-BASIC, ricordate di eliminare l'istruzione **List Funzione** (presnte, più o meno, a metà listato).

Quando inserite una vostra funzione (come argomento di DEF FN), fate in modo di esaminare dapprima il suo scheletro; in seguito, se la schermata sarà di vostro gradimento, tracciate l'intera superficie. Nel listato è inserita, a scopo dimostrativo, la funzione **z = Cos (x ^ 2 + y ^ 2)**.

Al termine del disegno si udirà un BEEP; la pressione di un tasto visualizzerà il listato, ponendo fine all'elaborazione. Per osservare un grafico in posizione ribaltata, basta cambiare segno alla funzione; nel nostro caso **z = - Cos (x ^ 2 + y ^ 2)**.

Non trascurate di tener conto delle istruzioni riportate a pagina 102 del manuale AmigaBasic e del fatto che alcune funzioni potrebbero non esistere in alcuni punti; tra queste **LOG (x)** ed **SQR (x)** non esistono per valori negativi di X.

Esempi di funzioni

SIN (x ^ 2 + y ^ 2)

Disegna in 3D, nei pressi dell'origine O (0, 0, 0), la funzione seno.

LOG (x ^ 2 + y ^ 2)

Disegna una funzione logaritmica che, nei pressi dell'origine, forma un "tunnel" a cuneo verso il basso. In prossimità dell'origine subisce un sobbalzo a causa della discontinutità nell'origine delle funzioni di questo tipo.

Si consiglia di osservarne dapprima lo "scheletro".

SQR (x ^ 2 + y ^ 2) * 6

Superficie che si "alza" e si "abbassa" diverse volte formando uno... spremiagrumi. Meglio evitare l'opzione scheletro, che genererebbe immagini confuse.

E' anche interessante osservare ciò che accade inserendo una funzione di una sola variabile. Con **x = 0**, ad esempio (e con **z = 4 * Cos (y) + 0**) verrà tracciata una superficie che ricorda la funzione **y = Cos (X)** in 2D.

Questi sono solo alcuni esempi, il resto è affidato alla vostra immaginazione.

Altri computer

La versione inviata dal nostro lettore è in **AmigaBasic**, compilabile con uno dei numerosi compilatori Basic disponibili per Amiga.

La leggibilità del programma potrebbe aumentare modificando (magari per puro esercizio) le linee che sono contraddistinte da un numero (**15, 20, 30** e 40).

Grazie alla notevole somiglianza di AmigaBasic con **Quick Basic** (il potente interprete / compilatore della **Microsoft**), dovrebbe esser decisamente semplice adattare il listato, qui pubblicato, all'ambiente **Ms-Dos**.

Come al solito, i lettori più intraprendenti sono invitati a scrivere la versione per altri linguaggi (oltre al Quick Basic, infatti, sarebbe interessante l'adattamento al **Turbo Pascal** e al "**C**"). Gli stessi lettori potrebbero approfittare dell'occasione per introdurre migliorie e incrementi vari di versatilità.

Inutile dire che i lavori migliori verranno pubblicati e, perchè no, compensati generosamente.

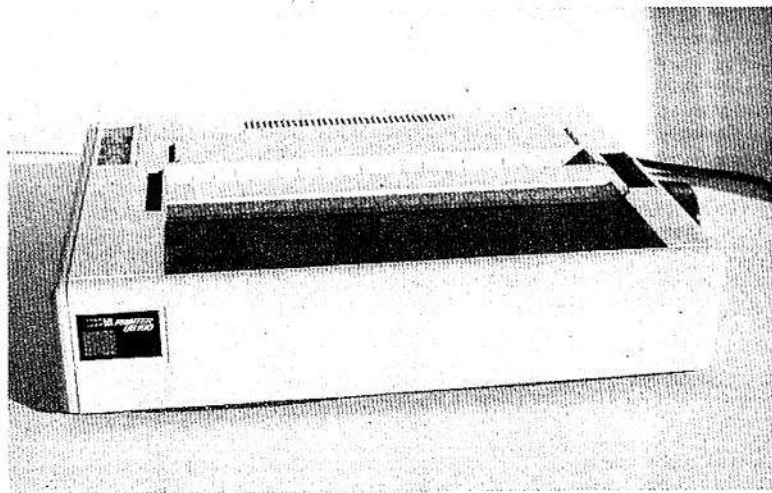
di Armando Sforzi

HARD COPY PER AMIGA 500

E' possibile sostituire il (macchinoso) programma GraphicDump disponibile in Workbench? La risposta è affermativa; anzi, chi lo desidera può apportare modifiche e migliorie

E' ormai universalmente riconosciuta la strabiliante capacità di Amiga nel gestire la grafica: il numero di colori utilizzabili, unito alla definizione dell'immagine ed alla velocità di tracciamento, rappresentano sicuramente il punto di forza principale del computer. Tuttavia a poco servirebbe il potenziale che ci ritroviamo sotto le dita se non ci fosse la possibilità di stampare quello che si riesce a produrre sul video. Il dischetto di **Workbench** mette a disposizione un programma scritto per questo scopo, **GraphicDump**, ma esso non è affatto di comodo utilizzo, specialmente se ci si trova all'interno di un programma in Basic. In nostro aiuto giunge allora l'utility che presentiamo in queste pagine, scritta interamente in **Assembler 68000** e realizzata con il diffusissimo pacchetto **Devpac, versione 2.14**, della HiSoft. Dal punto di vista strutturale, il programma è relativamente semplice, ma per comprenderlo facilmente riportiamo, a parte, una sua ipotetica "traduzione" in AmigaBASIC. Nelle prime due righe viene aperto il canale di comunicazione con la stampante e si regola l'interlinea; in seguito si aprono tre cicli nidificati grazie ai quali si "leggono" le porzioni della bit-map compresa nel riquadro di coordinate **(x0, y0) -(x1, y1)**; infine si chiude la comunicazione con la stampante, inviando il comando di reset generale (**Chr$ (27) "@"**). E' bene, ora, analizzare passo passo lo sviluppo del programma per scoprire particolarità sconosciute ai programmatori ad 8 bit (quelli del C/64, per intenderci).

Il disassemblato

Consideriamo, allora, il listato della routine ottenuto con **GenAm**, il potente programma disassemblatore di **DevPac**.Il primo comando è, di per sè, molto particolare: "**opt p+**". Che cosa significa? In realtà, per il microprocessore di Amiga, il comando non rappresenta nulla dal momento che si tratta di un codice utile solo per il programma assemblatore: si limita a controllare che, in tutto il segmento l.m., non vi sia alcuna istruzione dipendente in qualche modo dalla posizione in cui si trova memorizzato il programma, come ad esempio un **JMP** (oppure **BRA**) relativo ad una locazione particolare definita all'interno del programma stesso. Disporre di comandi compatibili con la rilocazione è una necessità inderogabile in quanto il Sistema Operativo dei computer moderni, a differenza dei vecchi **65XX** del C/64 (e simili),

gestisce in modo del tutto particolare la memoria a disposizione. I nuovi S.O., infatti, allocano files, dati e strutture, all'interno di aree Ram in cui risultino liberi spazi di memoria. Non sempre detti spazi si trovano sempre allo stesso indirizzo, nè è possibile prevedere l'area che verrà utilizzata dal programma. E' quindi necessario che gli indirizzi **assoluti** siano sempre costruiti in relazione al **Program Counter**: ecco spiegata la presenza del suffisso **(pc)**, presente in numerose istruzioni del listato che esigono un salto assoluto, oppure l'acquisizione di un in dirizzo effettivo o il suo contenuto. Detto questo, procediamo nell'analisi del disassemblato, in cui incontriamo una zona (linee 12 - 17) ove si definiscono le costanti valide per ottenere gli esatti indirizzi delle funzioni di libreria. Dalle linea 21 alla 39 vengono definite tre **"macro"** (o, se volete, etichette) che racchiudono, al loro interno, il codice macchina compreso tra i termini **macro** e **endm**. In pratica, il programma assemblatore, ogni volta che incontra un'etichetta macro, ricopia in memoria l'intero codice da essa rappresentato. Le macro, quindi **NON** sono subroutine; adoperarle non significa risparmiare memoria, ma esplicitano la propria utilità nella chiarezza e nell'ordine che portano in un listato.

La prima istruzione vera e propria si trova alla linea 43: **[MO VEM.l d0-d7/a0-a6,-(sp)]**. Questo comando fa sì che vengano memorizzati i valori di tutti i Registri interni del microprocessore; ciò è molto importante in quanto l'interprete di AmigaBasic, quando riprenderà il controllo delle operazioni, **DEVE** ripartire nelle stesse condizioni del momento in cui ha lasciato il controllo alla routine, pena inevitabili Guru Meditation. Una qualsiasi routine che debba funzionare insieme ad AmigaBasic, si deve sempre incaricare di salvare, **prima** di svolgere un qualsiasi altro compito, il valore dei Registri del 68000. Dalla linea 44 alla 61 vengono aperte le librerie contenenti le funzioni che verranno utilizzate dal nostro programma: **graphics.library, intuition.library e dos.library**. Dalla linea 66 alla 75 viene aperto il canale della stampante parallela ed inizializzata la periferica. Dalla linea 79 in avanti, il listato prosegue in stretta analogia con quello scritto in AmigaBasic prima visto, aprendo tre cicli e leggendo tramite la funzione **ReadPixel** il colore del punto individuato di volta in volta. C'è da notare che, per accelerare la velocità di esecuzione del programma, si fa intenso utilizzo dei Registri Dati (**d0 - d7**) del microprocessore. Dalla linea 130 inizia la routine di chiusura del programma, ove viene ripetutamente invocata la funzione di chiusura delle librerie. L'ultima istruzione prima di RTS, cioè **MOVEM.l (sp)+,d0- d7/a0-a6**, riassegna i valori originali a tutti i Registri, per non creare problemi all'Interprete di AmigaBasic quando gli verrà restituito il controllo. L'ultima parte del listato, come si vede, è occupata da uno spazio di memoria nel quale sono allocati dati, cioè stringhe e valori numerici, definiti con il comando **"dc.b"** (define constant.byte) e dove vengono assegnati alcuni spazi vuoti, lunghi ciascuno 4 byte, definiti tramite **"ds.l"** (define space.long). questi altro non sono che variabili, richiamabili tramite le eti chette che li precedono, nei quali si possono depositare valori da riutilizzare nel corso del programma.

Occupiamoci, ora, di come sia possibile ottenere il segmento l.m. che costituisce, in pratica, la routine vera e propria da utilizzare nei programmi in Basic.

Per i più bravi

Non potevamo certo farci sfuggire l'occasione per proporre ai lettori più in gamba la realizzazione di routines di hard copy più efficienti e versatili della procedura che compare in queste pagine.

Senza nulla togliere ai meriti del nostro collaboratore **Armando Sforzi** (che ha realizzato una routine di notevole utilità e, soprattutto, di rapida comprensione), l'algoritmo per riportare su carta ciò che appare su schermo può subire miglioramenti, anche sostanziali.

Ad esempio, si potrebbero inserire opportune correzioni in modo che si elimini, a seconda della stampante utilizzata, il fastidioso fenomeno della "compressione" dell'immagine, a causa del quale un quadrato visualizzato su video verrà riprodotto, su carta, come se fosse un rettangolo con il lato più lungo, a seconda dei casi, parallelo all'asse X oppure all'asse Y.

Interessante sarebbe anche la possibilità di selezionare, contemporaneamente, due o più finestre da riportare su carta.

Il trattamento cromatico (per chi, ovviamente, dispone di una stampante a co lori) sarebbe veramente l'optimum e dimostrerebbe di essere programmatori di notevole livello professionale.

Noi l'idea l'abbiamo data. A voi il (duro) lavoro per realizzare quanto suggerito...

```
OPEN "PAR:" FOR OUTPUT AS 2
PRINT #2, CHR$(27)"A"CHR$(8);
dy=(y1-y0)+1
FOR k = y0 TO y1 STEP 8
   PRINT #2, CHR$(27)"*"CHR$(3);
   PRINT #2,CHR$(dy MOD 256)CHR$(INT(dy/256));
   FOR j=x0 TO x1: b%=0
      FOR u=0 TO 7
         IF k+u <= y1 THEN
            IF POINT (j, k+u) = col THEN
               b%=b%+2^(7-u)
            END IF
         END IF
      NEXT
      PRINT #2, CHR$(b%);
   NEXT: PRINT #2, CHR$(13)
NEXT
PRINT #2, CHR$(27)"@"
CLOSE 2
```

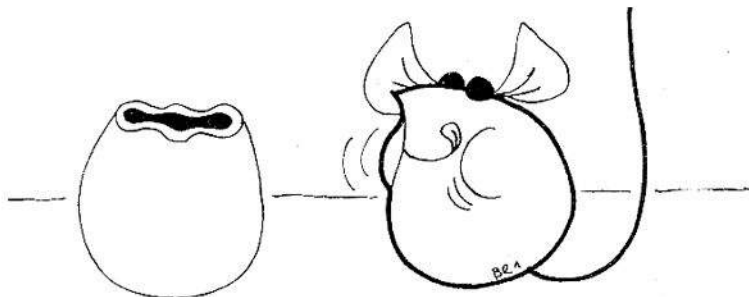
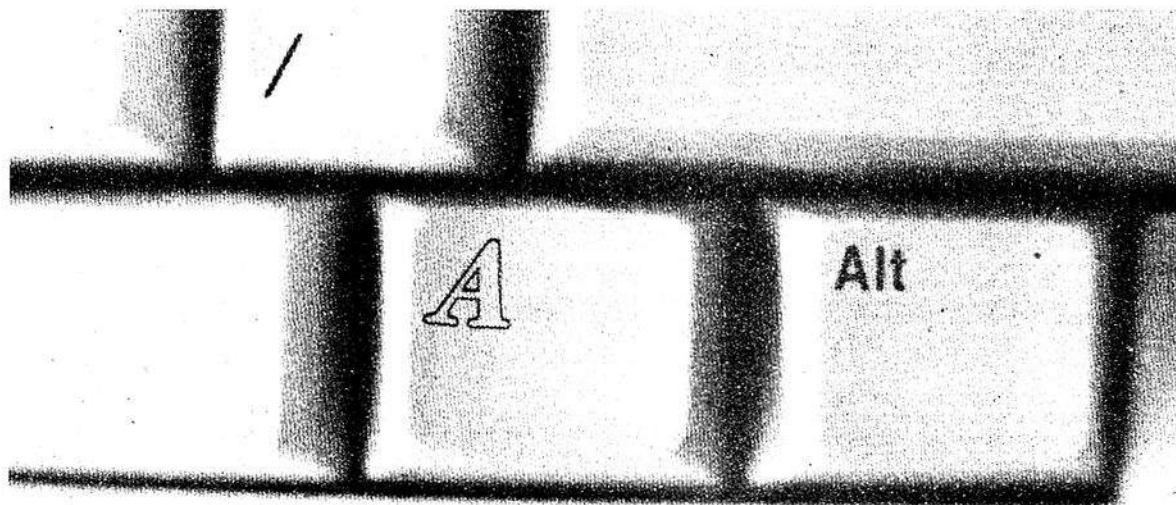
"Traduzione" della routine l.m.

```
  * HARD_COPY

  * ) Armando Sforzi  23.09.90  Amiga 500

  * chiamata da AmigaBASIC:
  *  CALL  ind&(rastport&,x0&,y0&,x1&,y1&,co-
lor&,mode&)


                 opt      p+           ;position
independent

  * define const.

  _LVOOpen            equ     -30         ;apre un
file
  _LVOWrite          equ     -48         ;scrive un
carattere
  _LVOClose          equ     -36         ;chiude un
file
  _LVOReadPixel      equ     -318        ;legge un
pixel
  _LVOOpenLibrary    equ     -552        ;apre una
libreria
  _LVOCloseLibrary equ     -414        ;chiude una
libreria

  * define macro

 CALL_FNLIB         macro               ;macro usa-
ta
                  move.l   \2,a6       ;per la
chiamata
                 jsr      _LVO\1(a6);delle fun-
zioni
                 endm                ;libreria

 CLOSE_LIB          macro               ;macro usa-
ta
                 move.l   \1,a1       ;chiudere
le librerie
                 jsr      _LVOCloseLibrary(a6)
                 endm

 PRINT              macro               ;macro usa-
ta
                movem.l d0-d7/a0-a6,-(sp);per
inviare
                  move.l   DOS_base(pc),a6;un
carattere
                move.l   \3,d1       ;alla stam-
pante
                 move.l   \1,d2
                 move.l   \2,d3
                 jsr      _LVOWrite(a6)
                 movem.l (sp)+,d0-d7/a0-a6
                 endm

  * open & initialize

            movem.l d0-d7/a0-a6,-(sp);salva il
valore dei
                                        ;registri
del 68000
                lea      GFX_name(pc),a1  ;indirizzo
libreria
                bsr      CALL_LIB         ;apre li-
breria
                tst.l    d0               ;errori?
                beq      Endlib1          ;se sì esce
                lea      GFX_base(pc),a0  ;idem per
le
                move.l   d0,(a0)          ;altre li-
brerie
                lea      INT_name(pc),a1
                bsr      CALL_LIB
                tst.l    d0
                beq      Endlib2
                lea      INT_base(pc),a0
                move.l   d0,(a0)
                lea      DOS_name(pc),a1
                bsr      CALL_LIB
                tst.l    d0
                beq      Endlib3
                lea      DOS_base(pc),a0
                move.l   d0,(a0)
                lea      rastport(pc),a0  ;salva va-
lore
                move.l   64(sp),(a0)      ;della ra-
stport
                lea      buff1(pc),a0     ;azzera
                move.l   #0,(a0)          ;il buffer1
                lea      Dev_name(pc),a1  ;indirizzo
del
                move.l   a1,d1            ;nome del
device
                move.l   #1006,d2         ;stato=non
c'è
                CALL_FNLIB Open,DOS_base(pc) ;apre
device
                tst.l    d0               ;errore?
                beq      Endfile          ;se siesce
                lea      Handle_out(pc),a0;se no,sal-
va l'handle
                move.l   d0,(a0)          ;di output
                lea      String_init(pc),a0;stampa la
stringa
                PRINT    32(sp),#3,Handle_out(pc) ;di
inizio stampa

  * main

                move.l   88(sp),d4
                lea      buff1(pc),a0
                move.b   d4,(a0)
                move.l   76(sp),d4        ;prepara
                sub.l    68(sp),d4        ;(y2-y1)+1
                addq.l   #1,d4            ;
                move.l   #$FF,d0          ;lo dispone
                and.l    d4,d0            ;nella for-
ma
                swap     d0               ;lo/hi
                or.l     d0,d4            ;e lo salva
                lea      buff1(pc),a0     ;nel buff-
er1
```

```
        or.l    (a0),d4
        move.l  d4,(a0)
        move.l  72(sp),d1
 loop1  lea     String_mode(pc),a0;stampa in
modo
          PRINT     32(sp),#5,Handle_out(pc)
;grafico
        move.l  68(sp),d2          ;carica
 loop2  clr.l   d3                ;i registri
per
        clr.l   d5                ;iniziare i
loop
 loop3  lsl.b   d5
        move.l  d1,d6              ;controlla
che
        add.l   d3,d6              ;il punto
non
        cmp.l   80(sp),d6          ;sia fuori
del
        bhi     .1                 ;riquadro
di stampa
        movem.l d1,-(sp)
        move.l  rastport(pc),a1  ;prepara
        move.l  d2,d0              ;i parame-
tri
        move.l  d6,d1              ;per la
funzione
          CALL_FNLIB ReadPixel,GFX_base(pc)
;di lettura/pixel
        movem.l (sp)+,d1
        cmp.l   84(sp),d0          ;colore
giusto?
        bne     .1
        ori.b   #1,d5              ;se sì lo
usa
 .1     lea     buff2(pc),a0       ;per co-
struire
        move.b  d5,(a0)            ;il carat-
tere grafico
 Endloop3 addq.l #1,d3            ;incrementa
il
        cmp.l   #7,d3              ;contatore
dei bits
        bls     loop3
        lea     buff2(pc),a0       ;stampa
        PRINT   32(sp),#1,Handle_out(pc) ;il
carattere
 Endloop2 addq.w #1,d2             ;continua
il loop
        cmp.l   76(sp),d2          ;per la
lettura
        bls     loop2              ;dei punti
orizzontali
        lea     String_lf(pc),a0 ;stampa
        PRINT   32(sp),#2,Handle_out(pc) ;a
capo
 Endloop1 addq.w #8,d1             ;continua
con
        cmp.l   80(sp),d1          ;nuova
striscia
        bls     loop1              ;di punti
        lea     String_reset(pc),a0;termina
la stampa
          PRINT     32(sp),#3,Handle_out(pc)
;col reset

 Endfile  move.l  DOS_base(pc),a6  ;chiude
          move.l  Handle_out(pc),d1;la comuni-
cazione
          jsr     _LVOClose(a6)    ;col device
di stampa
          move.l  4,a6
 Endlib3  CLOSE_LIB DOS_base(pc)   ;chiude le
librerie
 Endlib2  CLOSE_LIB INT_base(pc)
 Endlib1  CLOSE_LIB GFX_base(pc)
          movem.l (sp)+,d0-d7/a0-a6;ripristi-
na i
                                   ;registri
del 68000
          rts                      ;esce dalla
routine

 CALL_LIB move.l 4,a6              ;sottopro-
gramma
          clr.l   d0               ;per l'a-
pertura
          jsr     _LVOOpenLibrary(a6);delle li-
brerie
          rts

 * data field

 INT_name          dc.b     'intuition.libra-
ry',0
 GFX_name          dc.b     'graphics.libra-
ry',0,0
 GFX_base        ds.l    1
 INT_base        ds.l    1
 rastport        ds.l    1
 DOS_name        dc.b    'dos.library',0
 DOS_base        ds.l    1
 Handle_out      ds.l    1
 Dev_name        dc.b    'PAR:',0,0
 String_init     dc.b    27,'A',8,0
 String_reset    dc.b    27,'@',13,0
 String_lf       dc.b    13,10
 String_mode     dc.b    27,'*'
 buff1           ds.l    1
 buff2           ds.l    1

                 end
```

```
' Demo HARD_COPY

' Armando Sforzi per Amiga 500

CaricaLm:
OPEN "df0:prog.ass" FOR INPUT AS 1
lm$ = INPUT$ (LOF (1), 1)
CLOSE 1

Inizializzazioni:
ON MOUSE GOSUB ControlloMouse: MOUSE ON
ON BREAK GOSUB Fine: BREAK ON
SCREEN 1, 640, 400, 2, 2
WINDOW 1, "window 1", (0, 0) - (320, 100), 1, 1
GOSUB Display1
WINDOW 2, "window 2", (400, 30) - (600, 100), 1, 1
GOSUB Display2
WINDOW 3, "window 3", (100, 92) - (500, 250), 1, 1
GOSUB Display3

RivelazioneEventi:
WHILE 1
SLEEP 'Stop da tastiera / bottone sx del mouse
WEND

ControlloMouse:
IF MOUSE (0)THEN
WinNumber = WINDOW (0)
WINDOW CLOSE WinNumber 'chiude la finestra scelta
WINDOW WinNumber, , , 1, 1
'la riapre ponendola in 1^piano
GOSUB SceltaDisplay 'rinfresca la finestra scelta
rp& = WINDOW (8): mode& = 0 'prepara i parametri
IF WinNumber = 1 THEN 'per la routine Lm
x1& = 0: y1& = 0: x2& = 240: y2& = 7
col& = 0
ELSE

x1& = 0: y1& = 0
x2& = WINDOW (2) - 1: y2& = WINDOW (3) - 1
END IF
IF WINDOW (0) = 2 THEN col& = 3
IF WINDOW (0) = 3 THEN col& = 2
indlm& = SADD (lm$) 'indirizzo della routine Lm

IndirizzoPari:
WHILE indlm& / 2 <> INT (indlm& / 2) 'ridefinisce l'indirizzo
lm$ = lm$ + "": indlm& = SADD (lm$) 'se non e' pari
WEND
CALL indlm& (rp&, x1&, y1&, x2&, y2&, col&, mode&)
'routine di hard_copy
END IF
RETURN

SceltaDisplay:
ON WinNumber GOTO Display1, Display2

Display3:
PALETTE 2, 0, 1, 0
FOR j = 0 TO 50
a = RND (1) * 400: b = RND (1) * 140
c = RND (1) * 400: d = RND (1) * 140
LINE (a, b) - (c, d), 2
NEXT
RETURN

Display2:
FOR j = 0 TO 3
CIRCLE (100, 35), 30, 3, , , j
NEXT
RETURN

Display1:
PRINT "demo 'Hard_Copy' per Amiga 500"
'stampa in window 1
PRINT: PRINT "Clickare sulla finestra voluta
PRINT "per effettuare la stampa grafica.
PRINT: PRINT "In questa finestra
PRINT "verra' stampata la prima riga
PRINT "in reverse.
PRINT: PRINT "Terminare con [AMIGA - .]
RETURN

Fine:
WINDOW CLOSE 3 'libera la memoria
WINDOW CLOSE 2 'dalle strutture
WINDOW CLOSE 1 'definite
SCREEN CLOSE 1
END 'e termina
```

Innanzitutto digitate il listato con l'editor di Devpac, controllatelo e salvatelo. Quindi impartite il comando di assemblaggio con l'opzione di output **NIL**, per verificare che non vi siano errori.

A questo punto occorre rilanciare l'assemblatore, questa volta indirizzando l'output in memoria; poi si può procedere, come indicato nel riquadri, per creare il file l.m. che verrà utilizzato dal programma AmigaBasic.

Come realizzare il file l.m.

Tramite il comando di **debug**, otterremo sul video l'inizio del disassemblato del programma e ne approfitteremo per prender nota dell'indirizzo iniziale della routine. Tramite l'opzione "S" di **MonAm**, digiteremo il nome con il quale salvare il file ed i successivi indirizzi (iniziale e finale); nel caso, ad esempio, **"xxxxxx"** fosse l'indirizzo iniziale (espresso in esadecimale) si avrebbe...
xxxxxx, xxxxxx + 243
...in cui **243** è il valore della lunghezza (decrementata di 1), espressa in esadecimale, della routine. Nel nostro caso essa è lunga, appunto, 580 bytes (in esa: 244). Una volta creato il file, il programma può essere utilmente adoperato in qualsiasi programma scritto in AmigaBasic.La sintassi, da Basic, sarà...
CALL indlm& (rp&, x0&, y0&, x1&, y1&, col&, mode&)
...in cui:
indlm& è l'indirizzo iniziale di locazione della routine, ottenibile, ad esempio, come nel demo allegato, con l'istruzione SADD;
rp& è l'indirizzo della rastport della window attiva, ottenibile con l' istruzione WINDOW(8);
x0& - y0& sono le coordinate del vertice superiore sinistro del riquadro da stampare;
x1& - y1& sono le coordinate del vertice inferiore destro del riquadro da stampare;
col& è il colore del disegno che si vuol stampare;
mode& è il valore della modalità di stampa grafica del device parallelo. Riferendoci ad una stampante che simula una Epson, si hanno le corrispondenze riportate in tabella. Modificando il valore di "mode&" si può, dunque, ottenere una stampa in varie "riduzioni", particolare, questo, che incrementa le potenzialità della routine. E' importante notare che, nella sintassi di comando, tutte le variabili devono essere di tipo "lungo" (4 bytes di lunghezza), cioè comparire con il suffisso "&". Nel demo incluso nel listato, oltre alle procedure di caricamento e di lancio del programma stesso, compare un loop denominato: **IndirizzoPari**. Ha il compito di assicurare che l'indirizzo di partenza della routine sia pari, altrimenti non fa proseguire il programma. L'esigenza di tale controllo risiede nel fatto che le istruzioni del microprocessore devono sempre occupare un indirizzo **pari** in memoria, pena il blocco del computer con la comparsa di un **Guru n. 3** (indirizzo errato, appunto). In genere c'è solo il 50% di probabilità che una variabile inizi ad indirizzo pari. Il semplice loop, quindi, definisce ripetutamente la variabile stringa contenente i dati della routine finchè non incappa in un indirizzo di inizio pari.
Esistono metodi, alcuni utilizzanti funzioni di libreria, certamente più eleganti per ottenere lo stesso risultato, ma il nostro ha il pregio di essere semplicissimo.

Ricordiamo, in ultimo, che per ottenere un corretto funzionamento, la routine deve lavorare su una **finestra attiva** e posta in **primo piano.**

Modo	Punti per riga	(Commento)
0	480	(densità normale)
1	960	(doppia densità, velocità dimezzata)
2	960	(doppia densità, velocità normale)
3	1920	(quadrupla densità)
4	640	(video grafico I)
5	576	(grafica plotter (x:y=1:1))
6	720	(video grafico II)

Corrispondenze con printer tipo Epson

AVAIL

Con questo comando è possibile un controllo approfondito della quantità di **memoria** disponibile. Assolve, in pratica, alla stessa funzione svolta dall'indicazione che normalmente appare sulla barra di schermo del Workbench, ma con molta più flessibilità e dovizia di particolari.
Risulta indispensabile, inoltre, quando si opera in ambiente **shell** (o cli) senza l'ausilio del Workbench. Apparentemente stringato, le sue possibilità di applicazione vanno ben al di là di quanto lasci trasparire il manuale del **Dos 1.3** in dotazione al computer, come dimostrato da quanto espresso in queste pagine.
Sarebbe buona prassi, per chi predilige (o intende approfondire) l'ambiente Dos, rendere **residente** questo comando ad ogni boot di sistema (= inserire l'istruzione *Resident Avail* nel file Startup-sequence del disco di boot), in modo da disporne "perdite di tempo" dovute ai frequenti accessi al disco.

Avail. Impartirlo senza altri parametri, è probabilmente l'uso più frequente del comando. In questa modalità visualizza l'attuale condizione della **memoria Ram** di Amiga, specificandone tanto la disponibilità totale quanto quella parziale, ovvero limitata alla sola memoria **Chip** o sola memoria **Fast** (se presente).
Già questa indicazione può risultare molto utile non solo per prendere atto della memoria utilizzabile, ma anche (ad esempio) per stabilire se il modello Amiga di cui si dispone è dotato dei (relativamente) **nuovi chip Agnus** in grado di indirizzare un Mega byte di memoria cosiddetta Chip. Assumendo come presupposto che il Megabyte esista (ovvero che il proprio computer sia un A-2000 oppure un A-500 con almeno 512K di espansione), basterà infatti osservare le indicazioni sulla memoria fast e quelle sulla memoria Chip per trarne le logiche deduzioni.

AVAIL CHIP FAST TOTAL

Chip, Fast, Total; opzionali. Aggiungendo uno di questi parametri ad Avail, si ottiene un'informazione più "mirata", riguardante appunto la singola porzione di Ram del tipo specificato (chip, fast oppure totale). In questo caso, l'indicazione riguarderà soltanto la memoria libera, senza le altre specifiche ottenibili impartendo Avail da solo. Apparentemente di poca utilità, il loro uso consente, invece, applicazioni di rilievo se sfruttati nell'ambito di un batch file, come mostrato nell'esempio riportato a parte.

Gia' pubblicati

A partire dal n.75 della rivista si è cominciato ad esaminare in profondità il **Dos** di Amiga, rivolgendosi soprattutto (ma non solo) a chi si avvicina per la prima volta a questo computer. Chiaro che, con il succedersi degli appuntamenti, l'integrazione tra quanto trattato in precedenza e gli argomenti affrontati in queste pagine si fa sempre più stretta, anche se viene volutamente mantenuta la maggiore autonomia possibile.
Per quei lettori che iniziano solo ora a seguire la nostra rivista, è consigliabile (anche se non indispensabile) procurarsi i numeri a partire dal già citato 75: si disporrà così di un completo compendio da affiancare alla consultazione della manualistica, spesso fin troppo succinta nelle sue indicazioni.

n.75	sort	execute	cd
	n.76	direttive batch	ed
assign	caratteri speciali	if	break
copy	delete	skip	**n.79**
date	format	lab	which
dir	protect	quit	newshell e newcli
install	rename		ser: par: e prt:
path			con: e newcon:
search	**n.77**	**n.78**	nil: e raw:

Un esempio evoluto

Si è già detto che l'uso di **Avail** può risultare particolarmente vantaggioso se applicato ad un Batch File. Naturalmente nulla vieta che si possa semplicemente inserire una riga con l'unica istruzione Avail nel file-comandi: il conoscere l'ammontare della memoria libera, e di quella in uso, può spesso venir comodo.
Ma, grazie alla possibilità di effettuare raffronti tramite la condizione **If**, è implementabile qualcosa di più.
Per esempio, il condizionare l'esecuzione di un programma alla quantità di memoria libera presente.
Vediamolo in pratica, proponendo un file batch cui assegneremo come nome **MRun**, una evoluzione personalizzata del comando **Run**, strutturato come nel riquardo di questa pagina.
Intanto qualche istruzione preliminare. Il file, come ormai dovrebbe essere ovvio, va copiato con un qualunque editor che produca un file **Ascii** (in mancanza di meglio, può andar bene l'**Ed** incluso nel disco Workbench, descitto sul n. 78 della rivista). Poi, anche questo abbastanza ovvio, la directory **C** di sistema, ovvero quella del disco adoperato come boot, deve contenere tutti i comandi adoperati nel batch, in particolar modo **Run** ed **Execute**, ma anche **Echo**, **If**, e naturalmente **Avail**. Inoltre, deve esistere un assegnamento **Env:**.
Niente paura. Se ne ignorate il significato, è sufficiente che adoperiate il disco Workbench per lanciare Amiga, che provvede automaticamente ad assegnare questo device logico, del quale ci occuperemo meglio in altra sede.
Per la cronaca, serve comunque a sfruttare delle variabili di ambiente, un po' come nei linguaggi di programmazione veri. Ne vedremo in pratica l'applicazione.
Il nostro nuovo comando andrà adoperato con la sintassi...

Execute MRun numbyt nomeprg

Execute, come più volte rimarcato (vedi Amigafacile n. 76-comando **Protect**), può essere eliminato se si rende autoeseguibile lo **script** (il batch file, per intenderci) adoperando la forma **Pro tect Mrun +S.**

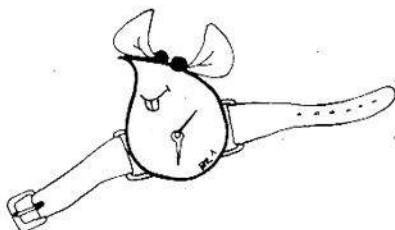
Il parametro **numbyt** indica la minima quantità di memoria libera necessaria affinchè un certo programma (**nomeprg**) venga eseguito. In pratica, se per esempio impartissimo...

Execute Mrun 100000 Editor

...verrebbe lanciato il programma di nome **Editor** come di consueto (tramite Run), ma **solo se** vi sono 100.000 byte liberi di Ram. In caso contrario, si otterrebbe solo una segnalazione della mancata esecuzione.
Escludendo l'aspetto didattico, il batch può risultare utile in altre occasioni.
Il file inizia con un semplice controllo che il primo parametro venga effettivamente inserito (tecnica più volte descritta), quindi entra subito nel vivo impartendo un **Avail Total** rediretto verso la variabile **mem**. Le variabili di ambiente, vengono memorizzate nel device **Env:**, associato (nel caso del Workbench) ad una directory omonima in Ram. Nel nostro caso, avremo un file di nome **mem** contenente il responso numerico di Avail. Uno dei pochi casi in cui è possibile adoperare queste variabili, è la condizione If. Ponendo il simbolo "$" davanti al nome della variabile (nel nostro caso Mem), la condizione farà riferimento al suo contenuto, ovvero il valore di memoria libera totale fornito da Avail Total.
Un po' complicato, ma in fondo non troppo, se si è abituati, per esempio, alle variabili del Basic.
Ecco allora che il batch confronta il valore da noi immesso con quello ottenuto da Avail (si noti il parametro **Val** affiancato ad **If**, descritto sul n. 77), ed esegue un **Run <nomeprogramma>** solo se la quantità di memoria disponibile è superiore a quanto da noi indicato.
Il batch può essere reso ancora più flessibile sfruttando, a seconda delle esigenze, i parametri Chip oppure Fast di Avail, per esempio quando si devono lanciare programmi grafici che, notoria mente, "succhiano" soprattutto memoria chip.

```
.key min,prog/a
if <min>X eq X
echo "Sintassi:"
echo "MRUN bytes nomeprog"
quit
endif
avail > env:mem total
if val $mem ge <min>
run <prog>
else
echo "Memoria totale inferiore"
echo "a <min> bytes!"
echo "Non eseguo il programma"
endif
```

Il file Batch di nome MRUN

DA AMIGA A MS - DOS

Il breve listato (scritto in **Gw - Basic**) sarà utile a coloro che sono soliti utilizzare programmi di word processor sia con l'Amiga che con sistemi Ms - Dos. E' noto che Amiga, infatti, alla fine di un rigo inserisce un **Chr$(10)** invece di un **Chr$(13) + Chr$(10)** come richiesto dalla maggior parte dei W/p in "ambiente" Ms Dos. Prima di usare il programma pubblicato, è indispensabile aver memorizzato i files di testo pervenuti, sul dischetto Ms - Dos, o tramite un programma di utilità (tipo **DosTo-Dos**) o direttamente, via interfaccia seriale. Il listato consente di visualizzare il contenuto di una qualsiasi directory Ms - Dos, di "convertire" fino a 5 files (appartenenti ad una qualsiasi Directory) e di memorizzare i files convertiti in una qualsiasi destinazione.

Inutile dire che il listato potrà rivelarsi utile anche come "base" di partenza per scrivere altri programmi di utilità che hanno l'esigenza di individuare uno (o più) file, appartenenti ad un qualsiasi percorso di directory, da memorizzare, dopo opportuno "trattamento" in altra parte di un dischetto.

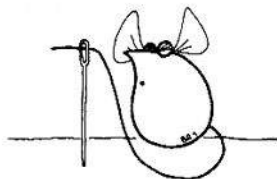
```
  90 ON ERROR GOTO 1000
  100 CLS : PRINT "Questo programma legge FINO A 5 files contenenti CHR$(10)";
  110 PRINT "e li trasforma tutti in CHR$(13)+CHR(10)"
  150 PRINT "Bisogna assegnare:": PRINT "1- Nome del drive sorgente": PRINT "2- Nome del
drive destinazione": PRINT "3- NOMI dei file sorgenti": PRINT "4- NOMI dei file destinazione":
PRINT
  160 PRINT "(Per interrompere la conversione premi un tasto)"
  170 RI$ = "": PRINT : PRINT "Vuoi esaminare la Directory?"
  171 INPUT "(Rispondi:  a,  b,  c, eccetera, oppure N)"; RI$
  172 IF RI$ = "" THEN 170
  180 IF RI$ = "n" THEN 210
  185 RI$ = RI$ + ":"
  186 PA$ = "": PT$ = "": INPUT "Nome eventuale path (\dir)"; PT$: PA$ = PT$
  187 IF PT$ = "" THEN PT$ = RI$ + "\"
  188 IF PA$ <> "" THEN PT$ = RI$ + PT$ + "\*.*": GOTO 200
  200 FILES PT$: GOTO 170
  210 INPUT "Nome drive sorgente     (a  b  c ecc.)"; SD$
  212 SD$ = SD$ + ":"
  215 INPUT "Nome eventuale path (\direc       ecc.)"; DS$
  216 IF DS$ <> "" THEN SD$ = SD$ + DS$ + "\"
  220 INPUT "Nome drive destinazione (a  b  c ecc.)"; TD$: TD$ = TD$ + ":"
  225 INPUT "Nome eventuale path (\direc       ecc.)"; DD$
  226 IF DD$ <> "" THEN TD$ = TD$ + DD$ + "\"
  230 INPUT "Quanti files da trattare"; NU: DIM N$(NU), T$(NU)
  233 PRINT : FOR NF = 1 TO NU
  236 PRINT NF; : INPUT "Nome file sorgente"; N$(NF): N$(NF) = SD$ + N$(NF)
  240 PRINT NF; : INPUT "Nome file destinazione"; T$(NF): T$(NF) = TD$ + T$(NF)
  242 PRINT : NEXT NF
  260 FOR NF = 1 TO NU: OPEN N$(NF) FOR INPUT AS #1
  270 OPEN T$(NF) FOR OUTPUT AS #2
  280 x$ = INPUT$(1, #1): IF EOF(1) THEN 1000
  285 B$ = INKEY$: IF B$ <> "" THEN BR = 1: NF = NU: GOTO 1000
  290 IF x$ = CHR$(10) AND DEV = 1 THEN 280
  295 IF x$ = CHR$(10) THEN PRINT #2, CHR$(13); CHR$(10); : PRINT CHR$(13): DEV = 1: GOTO
280
  300 DEV = 0: PRINT #2, x$; : PRINT x$; : GOTO 280
  1000 CLOSE #1: CLOSE #2: FOR NB = 1 TO NF: BEEP: NEXT NB: FOR NB = 1 TO 2500: NEXT NB:
PRINT : PRINT : PRINT : NEXT NF
  1001 IF BR = 1 THEN PRINT "ATTENZIONE!!! La conversione e' stata interrotta"
  1002 PRINT "Ho appena convertito i(l) file(s):": PRINT : PRINT
  1003 FOR NF = 1 TO NU: PRINT NF; N$(NF); TAB(25); T$(NF): NEXT NF
  1010 x$ = INKEY$: IF x$ = "" THEN BEEP: FOR B = 1 TO 2500: NEXT: GOTO 1010
  1020 RUN
```

JOIN

Join. Serve a "fondere" più files in uno unico, ma anche a realizzare rapidi **Append** da ambiente Shell. Teoricamente può essere applicato a qualunque tipo di file, compresi quelli binari (i programmi, insomma), ma in quest'ultimo caso il risultato è del tutto inutile, in quanto il file prodotto non potrebbe in alcun modo essere mandato in esecuzione. Il suo campo di applicazione riguarda, dunque, principalmente i files **Ascii**.

File1 ... File n. Sono i nomi dei files che si intendono fondere, per un massimo di **15**. I nomi devono essere divisi tra di loro da uno **spazio**, e possono includere anche il loro **path**, ovvero la specifica di unità, drive, directory, eccetera. Il che implica, come ovvio, che non è indispensabile si trovino tutti nella stessa directory, ma, anzi ,si può utiizzare il comando per fondere files tratti da più directory in uno presente in quella corrente, o addirittura in una ulteriore "esterna", purchè si fornisca di path anche il file destinazione.

Impartito da solo, non sortisce alcun effetto, se non quello di un lapidario **Bad Args.**

JOIN file1 file n TO newfile

To. Keyword da specificare sempre, eventualmente sostituibile con AS. Pura questione di gusti.

Newfile. Il nome del file destinazione, che conterrà il testo di tutti i file sorgente, nell'ordine in cui sono stati elencati.
Per avere un esempio pratico, si provi ad editare tre files adoperando direttamente la Shell, ovvero seguendo questa procedura:
* Si impartisca...
Copy * to ram:t1
* Si digitino un paio di parole qualsiasi (come **prova 1**), poi si prema il tasto Return, quindi **Ctrl+** \ (barra inversa). Così facendo (tecnica più volta affrontata in Amigafacile), si avrà in Ram Disk un file testo di nome **t1**, contenente il fantasioso messaggio *prova 1*.

* Si ripeta l'operazione altre due volte, cambiando il nome del file in **Ram:t2** e **Ram:t3**, ovviamente digitando nel testo un numero di prova analogo.

* Infine si impartisca **Cd Ram:** (giusto per evitare lunge digitazioni) e ad eseguire...

Join T1 T2 T3 To Test
Per vedere il risultato, non resta che un banale **Type ram:test.**

C'è inoltre da dire che, volendo, anche la procedura prima descritta per ottenere un file Ascii, ovvero la redirezione della console corrente (il simbolo "*"), può essere modificata.
Nelle nostre precedenti dissertazioni, abbiamo infatti fatto più volte uso di questa tecnica, adoperando con egual disinvoltura **Copy** oppure **Type**. Ebbene, a questi può essere aggiunto anche il nostro **Join**! E' possibile, infatti, adoperare un comando abbastanza simile a quelli già noti...
Join * To Ram:test
...per redirigere la console verso il file di nome test memorizzato in ram, adoperando sempre **Ctrl+** \ per concludere le operazioni.
Mentre, però, con Copy e Type verrà sempre sovrascritto un eventuale file dello stesso nome, Join consente un'alternativa. Con...
Join * to ram:test
... si avrà in tutto e per tutto lo stesso risultato ottenibile con Copy e Type. Adoperando però ad esempio...
Join * ram:t1 To Ram:test
...si potrà editare un testo che verrà accodato al file **T1** (un vero e proprio append), ed il tutto verrà poi salvato nel file di nome Test.

ALIAS

Una volta tanto, ci si occupa un po' di un comando... che **non** è un comando. Inutile passare al setaccio le varie directory C, System, Utilities, eccetera del disco Workbench: non lo si troverà. Per reperirne qualche traccia, nonchè un esempio di come va adoperato, si può leggere con Type il file **Shell-Startup** presente nella directory **S** del disco Workbench. Già il nome del file, che contiene la serie di "alias", ci fornisce una soluzione all'amletico dilemma sulla sua esistenza: esiste, ma è un'applicazione *intrinseca* alla Shell. Il che implica, tra l'altro, che può essere sfruttato solo se si adopera Shell, **non** Cli.
Nel caso lo si intenda adoperare nell'ambito della startup-sequence, risulta quindi indispensabile che sia prima attivata Shell, il che sottindende (si veda con Type il file di startup incluso nel disco Workbench e la descrizione su Amigafacile n. 79) il **"mount"** del device **Newcon:**. Dal punto di vista funzionale, Alias fa esattamente ciò che ci si aspetta dal termine, sinonimo di **"soprannome"** o qualcosa del genere: affibbia ad uno specifico comando un altro nome.

Ma in realtà consente molto di più.

Nome. Questo primo parametro indica il nuovo nome con il quale verrà riconosciuto un determinato comando.
Non esistono limiti alla fantasia, tuttavia è buona norma non discostarsi molto da quello originario o dalla funzione svolta dal nuovo comando, se si è operato in maniera "massiccia" sul secondo parametro (vedi descrizione).
Nell'uso più banale di questa risorsa, ci si può ad esempio limitare ad un semplice "accorciamento" dei nomi di alcuni comandi, al nobile scopo di premiare l'innata pigrizia di noi utenti. Se, quindi, impartire (p. es.) **Loadwb** costituisce una mole di lavoro troppo gravosa, ecco che con...
Alias WB Loadwb
...il problema è risolto: ogni qualvolta si vorrà aprire uno schermo Workbench, lo si potrà fare adoperando **WB** invece di **Loadwb**.
Naturalmente la forma originaria resterà comunque attiva.

ALIAS nome comando

Comando. La seconda specificazione di Alias precisa il comando che verrà eseguito ogni volta che si adopererà, in ambiente Shell, il nome assegnato al primo parametro. Già detto di un uso semplificato di Alias, aggiungiamo ora che l'attribuzione del parametro non si limita al **nome** di un comando, ma all'**intero** comando, comprensivo cioè di suoi eventuali parametri. Ecco dunque dove si rivela la potenza di Alias. Nella descrizione del paranetro **<nome>** abbiamo usato come esempio Loadwb, che tramite Alias diventava un più comodo Wb. Si può però estendere l'esempio con un più proficuo:
ALIAS Wb Loadwb -debug
Stavolta la comodità diventa decisamente più palpabile, in quanto si eviteranno noiose digitazioni. Invocando WB da Shell, verrà aperto uno schermo Workbench comprensivo del menu nascosto (opzione **-debug**), notoriamente utile soprattutto per recuperare un po' di memoria dopo l'esecuzione di un'applicazione (opzione **flushlibs**).
Come intuibile, grazie alle prestazioni di Alias è quasi possibile creare dei nuovi comandi. Un esempio tipico è proprio riportato nella shell-startup del disco Workbench, che abilita un comando **Clear** per svuotare lo schermo assegnando, come secondo parametro, **Echo** seguito da una serie di codici Ansi per raggiungere lo scopo.
C'è da aggiungere ancora un particolare: il secondo parametro di Alias deve contenere obbligatoriamente un comando raggiungibile dal Dos, tipicamente inserito nella directory **C**, o comunque in una delle directory incluse nel percorso di ricerca. I parametri del comando possono essere specificati espressamente nell'invocare Alias, ma possono anche essere demandati ad una precisazione futura sotituendoli con due parentesi quadre così configurate: "**[ ]**". Con ciò, si aumenta ancora di più la flessibilità dello pseudo-comando. Un esempio banale, ma non troppo. Come già accennato nella trattazione del comando Format, questo è fornito di una sintassi decisamente prolissa. Con Alias è possibile renderla più umana, per esempio con...
Alias F1 Format Drive Df1: Name MIO noicons
Rimandando al n. **76** per una completa disamina del programma **Format**, con questo comando si otterrà che impartendo solo **F1** verrà formattato un floppy inserito nel drive **df1:** (volendo, si possono predisporre **due** alias, uno per df1: ed uno per df0: semplicemente cambiando Drive Df0: nella sintassi ed assegnando **F0** al nuovo alias).
E già la cosa può risultare abbastanza comoda. Tuttavia, tutti i dischi formattati adoperando F1 avranno lo stesso nome (nell'esempio: **MIO**). Adoperando invece una forma...
Alias F1 Format drive df1: Name [ ] noicons
...ecco che si avrà un comando alias più completo: impartendo **F1 Pippo** si formatterà un disco con nome Pippo, con F1 Vattelapesca il disco avrà nome Vattelapesca, e così via.
In definitiva, un Alias da sfruttare a più non posso.

GENERATORE MUSICALE

Siete stonati come una campana e non vi riesce, quindi, di scrivere musica per la vostra amata? Siamo qui per aiutarvi...

C'era una volta, tanto tempo fa, un sistema computerizzato chiamato MSX...

Scherzi a parte, la definitiva uscita di scena del sistema **MSX** (che doveva costituire, nelle intenzioni dei suoi progettisti, una specie di anello di congiunzione tra un sistema - giocattolo ed il sistema Ms - Dos) non lascia rimpianti, ad eccezione di un paio di occasioni mancate.

La più importante di queste, secondo noi, era costituita da una fantastica (per quei tempi) cartuccia Rom che, inserita nel computer, traformava la tastiera dei caratteri un una tastiera musicale.

Tutto qui? -direte voi-

Un momento - risponderemo noi- Non si trattava di uno squallida riproduzione di uno dei tanti programmi per Vic 20 che offrivano opportunità di infimo ordine musicale, ma di una vera e propria novità in campo informatico.

Ad ogni tasto della tastiera corripondeva una nota musicale: premendo il tasto **A** sentivate un **Do**, ad **S** corrispondeva il **Re**, e così via. Se, però, cercavate di suonare in sequenza due note discordanti tra loro, il programma di gestione cercava la nota musicale che, in quel momento, era **la più vicina** alla nota corrispondente al tasto **errato** e la emetteva.

Non suonavate, ovviamente, la nota che (ripetiamo, per errore) volevate, ma il risultato era una melodia straordinaria, priva di errori musicali, almeno per chi non si intende di musica e non è (mai stato) in grado di suonare uno strumento musicale.

Ricordo di aver dedicato molte ore all'uso del programma: bastava premere tasti a casaccio per ricavare una musica semi - casuale niente male; era poi sufficiente inventarsi un paio di rime tipo "cuore e amore" per avere ai miei piedi grappoli di ragazze febbricitanti d'amore...

Forse non andò proprio così (mi riferisco alle conquiste, naturalmente) ma la colpa non può certo essere attribuita al software nè al Sistema.

La nostra sfida

Per i meno svelti di comprendonio, precisiamo che la sfida di questo mese consiste nello scrivere un programma in grado di far corrispondere alla fila di tasti A, S, D... L (seconda fila dal basso) le note Do, Re, Mi... Alla terza fila, simulando i tasti neri del pianoforte, dovranno corrispondere le note musicali opportune.

E fin qui, nulla di speciale, ovviamente.

Il nucleo più importante del software che vi suggeriamo di scrivere avrà il compito di **gestire i tasti "sbagliati"**. Ciò significa che se, dopo aver suonato una certa nota, quella che stiamo per suonare non genera "stridore" (di tipo logico, ovviamente) sonoro, viene suonata normalmente. Se, invece, viene riscontrata una stonatura, viene subito individuata (in modo possibilmente **non** casuale, ma determinato da un algoritmo) la nota più adatta alle circostanze e suonata dopo aver segnalato, su video, l'"errore" sia commesso, sia la correzione apportata.

I computers

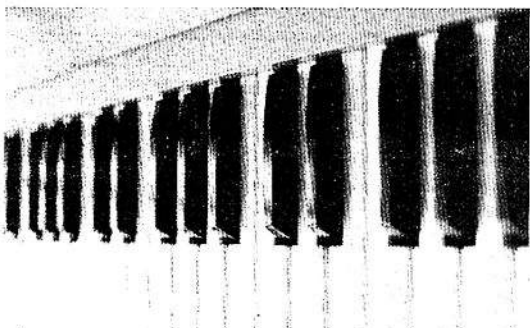
Verrà data precedenza ai programmatori che posseggono computer **Amiga** oppure **Ms - Dos** compatibili (utilizzati con *qualsiasi* linguaggio di programmazione). Gli utenti del **C/64** (dato che il programma non ha la necessità di sfruttare particolari caratteristiche hardware della macchina) sono egualmente invitati a partecipare alla sfida, purchè il lavoro eseguito venga inviato su dischetto (verranno inesorabilmente cestinati nastri-cassette eventualmente pervenuti).

Concentratevi sull'algoritmo di individuazione (e relativa correzione) dell'"errore" di battitura. Lasciate perdere virtuosismi sui colori di visualizzazione dei tasti o sulle note stesse (effetto eco, vibrato, selezione e gestione ADSR e così via). A complicare le cose ci penseranno i lettori, dopo aver digitato il listato il listato che pubblicheremo sulla rivista.

Inutile dire, quindi, che più breve sarà il programma, più elevate saranno le probabilità che sia pubblicato. Il migliore listato verrà premiato con **200 mila lire** che, con i tempi che corrono, non è poi così poco.

Per informazioni, telefonate in Redazione il giovedì pomeriggio: 02 / 57. 60. 63. 10

POSTAMIGA

(a cura di A. de Simone)

Bianco candido

Ho un problema di stampa su carta quando uso un programma grafico come Deluxe Paint. Dopo aver riempito lo schermo di bianco e disegnato qualche oggetto con più colori, in fase di stampa, sullo sfondo bianco, appaiono dei puntini. Come posso fare per non fare stampare nulla sul bianco?

(Sergio Giuffrida - ?)

Intanto, come giustamente arguisce il lettore nel prosieguo della lettera, per avere i colori espressi in tonalità di grigio è necessario avere settato da **Preferences/Graphic1** l'opzione **Gray Scale**. Se non si ha bisogno di più tonalità, una prima soluzione al problema potrebbe proprio essere quella di eliminare i grigi, optando per "**black and white**" (no, non si sta parlando di alcolici...).

In ogni caso, una vera soluzione necessita di una corretta determinazione dei colori nel programma grafico. Il che vuol dire, innanzitutto, prestare attenzione alla definizione **rgb** del bianco, che deve essere ottenuto settando alla massima intensità le tre le componenti cromatiche della palette colori.

Può capitare, infatti, che quello che noi vediamo come bianco sia, in realtà, un grigio molto chiaro, che però viene rilevato in fase di stampa e quindi *"puntinato"*.

Con tutta probabilità, però, il problema, nel caso specifico, è legato al modo di ottenere il colore di sfondo.

Dalla lettera traspare infatti l'uso del cosiddetto **fill** per "riempire" di bianco lo schermo, cosa che può facilmente provocare qualche lieve imprecisione, magari dovuta a residui di altri colori molto chiari non visibili ad occhio nudo.

Per evitare simile eventualità, ed essere in ogni caso più sicuri del risultato, è preferibile adoperare i cursori dell'opzione **Color Palette** (o similare, a seconda del tool grafico adoperato) per modificare definitivamente il colore di sfondo in modo da fargli assumere il bianco.

Ansi, Basic e modem

Ho letto su Postamiga n.78 dell'uso dei codici Ansi, che ho subito usato nella mia startup-sequence. Mi chiedo però se sia possibile utilizzarli in Basic, al posto di Color e Palette.

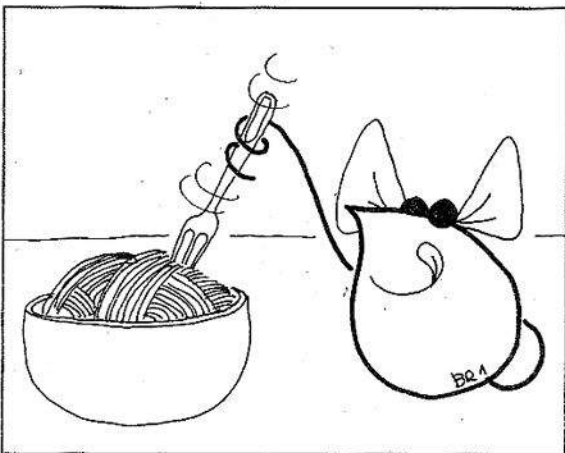
Ho provato in tal senso, ma l'unico risultato è stato un bel mal di testa.

Inoltre, vorrei sapere come usarli per un mio programmino di gestione del modem, per il quale ho preso spunto dai vostri consigli sulla rivista.

(Giuseppe Burgio - via BBS)

Per i mal di testa di questo genere non c'è che un rimedio: un analgesico.

Scherzi a parte, i simboli **Ansi** non possono essere utilizzati direttamente da Basic con qualche **Print**, tantomeno adoperando **Color** e **Palette**, peraltro già sufficienti per comuni applicazioni.

Un metodo per ottenere qualcosa di simile agli effetti dell'Ansi in Basic, limitatamente agli **stili di scrittura**, è stato comunque pubblicato su

Postamiga del n. **75** ("Questione di stile"). In quella sede, si utilizzavano alcune funzioni della libreria grafica per ottenere lo scopo.

Volendo rendere più flessibili le capacità di scrittura su video, implementando qualcosa di paragonabile al codice Ansi, si dovrebbe ricorrere ancora più massicciamente alle librerie di sistema, accedendo alle modalità di Input/Output della **Console.Device** di Amiga.

Il che significherebbe aprire la suddetta device e quindi manipolarla. Compito, questo, irto di difficoltà, e decisamente sconsigliato per chi adopera solo il Basic, senza conoscenze più che approfondite su come "si muove" il sistema.

Basti pensare che, solo per l'accesso ad un device, è necessario spremere a fondo la libreria **Exec**, adoperando Funzioni come **AllocSignal, Dolo, Addport**, e chi più ne ha più ne metta.

Ognuna di queste, poi, richiederebbe da Basic una manipolazione a suon di **Poke** delle strutture eventualmente ad esse collegate, la cui identificazione passa necessariamente attraverso uno studio più che approfondito dei manualoni della **Addison Wesley**. Capita l'antifona?

Niente che non si possa fare, per carità, ma una descrizione sul come procedere non è certo adatta a queste pagine; se ne riparlerà, eventualmente, nelle sezioni più "toste" della rivista, dedicate alla programmazione in **C** o **Assembly**.

Anche in questa eventualità, comunque, è bene tenere sotto mano un analgesico...

Quanto all'invio dei codici Ansi attraverso un programma di comunicazione, la cosa è più semplice di quanto si creda. Supponiamo, per esempio, di inviare via modem una schermata Ansi sul monitor di chi è collegato con noi attraverso la linea telefonica (metodo da evitare, però, se si tratta di una bbs!).

Anzitutto bisognerà editare un file adoperando una delle tecniche descritte sul n. 78. Questo, in fin dei conti, non sarà altro che un file Ascii contenente anche simboli atti a generare diversi **colori**, o **stili** di scrittura.

In quanto tale, basterà che il programma di gestione lo apra, ne prelevi i caratteri byte per byte, e li invii al device **Com1:** preventivamente inizializzato. In alternativa, se non è troppo lungo, si può caricare in una stringa tutto il contenuto del file, e quindi inviarlo a **Com1:**.

Rinviando al listato **Miniterminal** pubblicato su Postamiga n. 75 per maggiori dettagli su come impostare un programma di comunicazione generico, la cosa può essere così concretizzata:

```
OPEN "com1:2400,N,8,1" AS #1
OPEN "Demo" FOR INPUT AS #2
WHILE not LOF(2)
a$=INPUT$(LOF(2),2)
PRINT #1, a$;
WEND
CLOSE
```

La velocità in baud (**2400**) andrà naturalmente modificata in base alle proprie esigenze, mentre in questo esempio si presuppone l'esistenza nella directory corrente di un file a nome **Demo**, contenente il testo frammisto ai codici Ansi.

Perchè l'invio funzioni, in remoto (= all'altro capo della linea telefonica) deve naturalmente essere attivo un programma di comunicazione in grado di emulare un terminale Ansi.

Un altro aspetto cui prestare attenzione, riguarda anche il computer adottato dal "ricevente".

Se non si tratta di un altro Amiga, possono sorgere problemi di compatibilità non solo a livello di Ascii (soprattutto per i caratteri di line feed e ritorno carrello, nonchè per tutti i codici ascii superiori a **127**), ma anche in rapporto a certe particolari sequenze che implementano p.es. il **neretto**, il *corsivo* e così via.

Come se non bastasse, l'esempio prima visto può andar bene per inviare un file Ascii / Ansi, ma se si dovesse prevedere, per un proprio programma, anche la visualizzazione di eventuali codici in ricezione.... si tornerebbe ai problemi prima accennati a proposito del Basic, anzi di AmigaBasic.

L'emulazione di un terminale, insomma, non è proprio uno scherzo. Inviare una schermata Ansi può anche essere semplice, ma un vero terminale Ansi è tutt'altra cosa. E a un livello di programmazione tale che, se anche lo si fosse raggiunto, sarebbe più opportuno sfruttarlo con linguaggi più adeguati.

Quale Assembler?

Vorrei cominciare a programmare in Assembler, e intendo procurarmi un assemblatore adeguato. Quale mi consigliate in rapporto alle prestazioni: il Seka, il DevPac, o qualche altro? Potreste dirmi anche se con un drive da 5.1/4 per Amiga ed il C/64 Emulator è possibile caricare i programmi registrati con il drive 1541?

(Marco Pedrutelli - Vimercate)

Esprimere un giudizio comparativo tra diversi pacchetti software è sempre problematico.

Nel caso specifico, tra i programmatori Assembly si trovano degli entusiasti del Seka, ma anche "irriducibili" del DevPac o di altri assemblatori come **AssemPro**, **ArgAsm**, eccetera.

Un parere, quindi, non può che essere soggettivo. Il problema, in realtà, si pone solo in rapporto a chi muove o intende muovere i primi passi in questo ostico linguaggio: una volta acquisita una certa padronanza, l'uso di uno, piuttosto che l'altro, diventa un semplice fatto di scelta personale.

Ma chi comincia, in genere, vuole proprio una risposta precisa, ed a ragione. Inevitabile, dunque, sbilanciarsi: meglio il **DevPac**, e per più di un motivo.

Intanto perchè ne esiste la **versione italiana**, ed il capire bene la documentazione è di primaria importanza.

In relazione alle prestazioni, c'è da dire che, rispetto al Seka, "educa" maggiormente ad una corretta programmazione, con la possibilità di

Emulare, che passione

Esiste compatibilità tra i file Amiga e quelli McIntosh? Amiga legge i floppy formattati Macintosh? E' effettivamente gestibile il megabyte di memoria che il produttore della scheda Pc Power Board dichiara?
(Bruno Masera - Torino)

Tra tanta emulazione, viene da chiedersi se il nostro lettore ha intenzione di adoperare Amiga, ogni tanto (si scherza, si scherza...).

La compatibilità tra file di diversi standard, si tratti del **Mac** come dei Pc **Ms-Dos**, in senso stretto **non** può esistere altro che a livello di file Ascii, e limitatamente ai caratteri con codice inferiore a 127. Anche in questo caso, comunque, c'è poi il problema di "trasferire" i files da un ambiente all'altro. Tutti i file che coinvolgono il sistema operativo (e quindi i programmi), non possono che essere compatibili con il proprio sistema, e quindi non utilizzabili in altri ambienti. A meno che, naturalmente, non si adoperi un emulatore software o hardware.

Questo, oltre che rendere possibile l'esecuzione di programmi del sistema che emula, deve anche occuparsi della gestione delle periferiche, in special modo dei **drive**, unico mezzo (se si esclude un collegamento via terminale) per rapportare i due mondi.

Il che comporta, ovviamente, che dei floppy formattati secondo i dettami dello standard "alieno" possano esser letti da Amiga: sfruttando direttamente i suoi stessi drive, o talvolta utilizzando drive del formato emulato. Così, ad esempio, un emulatore software Ms-Dos come il **Transformer** utilizza i drive Amiga, e lo stesso fanno le schede hardware **Pc Board** e **ATonce** per Amiga 500. Come esempio contrario, l'emulatore Macintosh **Amax** necessita invece di un drive di quel formato.

In entrambi i casi, comunque, la leggibilità dei file è assicurata, in genere con ottima compatibilità di esecuzione.

Quanto alla memoria della scheda PcBoard, peraltro recensita proprio sul numero scorso della rivista, questa viene effettivamente "ceduta" ad Amiga quando lo si usa in modalità standard, per un ammontare di 512 KB di Fast Ram, più altri 512 KB utilizzabili come Ram Disk resistente al reset.

sfruttare senza troppo sforzo le risorse del sistema: il Devpac implementa files di inclusione, il Seka no (anche se, in teoria, si può crearli da sè).

Ci sarebbero anche altri motivi, ma troppo tecnici per chi è ancora in fase di "accostamento" all'Assembly. Detto questo, comunque, non va negata l'utilità del Seka nello stilare brevi routines: in questo caso, la sua velocità di assemblaggio può risultare davvero comoda.

Per un vero e proprio "programma", però, è opportuno rivolgersi a qualcosa di più valido.

Quanto alla seconda domanda, la risposta sta tutta in un solo monosillabo: NO.

CMD di poco uso

Sono un utente di Amiga 500, e le mie conoscenze non sono troppo a livello terra-terra.
Non ho ben capito, però, a che cosa serva il programma CMD presente nel disco Workbench 1.3.
(Antonio Artale - Siracusa)

Cmd, contenuto nella directory **Utilities** del disco Workbench, è in pratica un comando di **redirezione** un po' particolare.

Nel senso che funziona con modalità opposte a quelle cui si è comunemente abituati smanettando con il Dos.

Si limita, come specificato nella criptica descrizione riportata sul manuale del Dos 1.3, ad intercettare l'output di un qualunque programma rivolto alla porta parallela o seriale, e redirigerlo verso un file.

Per default questo file assume il nome **Cmd_file**, e viene memorizzato nella **Ram disk**. Il nome, come pure altri parametri riportati sul manuale in dotazione al computer, può essere modificato agendo sulla **Info** dell'icona associata al programma, o tramite opportuni parametri aggiunti se si invoca Cmd da Shell o Cli.

Il modo più semplice per comprendere come funziona, è ricorrere ad un esempio pratico. Si lanci il sistema col disco **Workbench**, quindi si apra l'icona della directory Utilities del disco in questione. Un doppio click su Cmd, e la redirezione è attivata.

Per vederla all'opera, si lanci ora il **Notepad**, e si scriva qualcosa all'interno della sua finestra.

Ora si proceda come se si volesse stampare su carta il documento: si selezioni **Print As/Draft**, quindi **Print/Autosize** dal menu **Project** del Notepad. Sullo schermo apparirà una finestra con l'indicazione dell'avvenuta redirezione, che subito dopo segnalerà la rimozione del Cmd. Tutto qua.

Per andare a vedere cos'è successo, è necessario, ora, scomodare la Shell, in quanto il **Cmd_file** non viene provvisto di icona. Si biclicki dunque su Shell, quindi si impartisca...

Type Ram:Cmd_file

Si potrà constatare come il contenuto del documento Notepad sia stato salvato in Ascii, anche se assieme ai codici di controllo del text editor (se proprio vogliamo chiamarlo così...).

Come ovvio, l'utilità di Cmd è decisamente limitata, a meno di applicazioni molto particolari: qualunque programma che utilizzi la porta parallela o seriale, normalmente consente anche il salvataggio del relativo file su periferica. Se per un qualunque motivo ciò non fosse possibile, ecco che Cmd può supplire alla necessità.

Requester facile

Ho un Amiga 500, e premetto subito che sono un (quasi) principiante alle prime armi. Ho cominciato a fare qualche programmino in Basic, ed ora vorrei lanciarmi in qualcosa di più impegnativo.
Potreste darmi un consiglio su come ottenere delle richieste cui rispondere col mouse (requester), ma senza troppe complicazioni (tipo librerie) che ancora non so adoperare?
(Stefano Corsari - Firenze)

Detto fatto. Un requester, in fondo, può benissimo essere emulato in "basic-basic" adoperando una finestra apposita, e controllando il movimento del mouse per stabilire se è stato adoperato il suo

```
' DEMO DI REQUESTER SENZA ADOPERARE LIBRERIE DI SISTEMA
'
a$ = "RICHIESTA DI INTERRUZIONE": b$ = "VUOI PROPRIO SMETTERE?"
c$ = "SICURO!": d$ = "NO!!!!": requester a$, b$, c$, d$, risp
IF risp = 0 THEN PRINT "NO" ELSE PRINT "SI"
END
'
' SOTTOPROGRAMMA "REQUESTER" ADOPERABILE IN QUALUNQUE LISTATO
'
SUB requester (riga1$, riga2$, scelta1$, scelta2$, risposta) STATIC
WINDOW 5,, (0, 0) - (311, 45), 16: largmax = INT (WINDOW (2) /8)
rig1 = LEN (riga1$): rig2 = LEN (riga2$)
PRINT: PRINT PTAB ((311- (rig1*8)) /2); LEFT$ (riga1$, largmax)
PRINT: PRINT PTAB ((311- (rig2*8)) /2); LEFT$ (riga2$, largmax)
scelta1$ = LEFT$ (scelta1$, 12): scelta2$ = LEFT$ (scelta2$, 12)
riq1 = (LEN (scelta1$) + 2) *8: riq2 = (LEN (scelta2$) + 2) *8
x1 = (312- (riq1 + riq2)) /3: x2 = x1 + riq1: x3 = x1 + x2: x4 = x3 + riq2
LINE (x1, 36) - (x2, 50), 2, bf: LINE (x3, 36) - (x4, 50), 2, bf
COLOR 1, 2: LOCATE 6, 1: PRINT PTAB (x1 + 8); scelta1$;
PRINT PTAB (x3 + 8); scelta2$;: COLOR 3, 1
WHILE -1
WHILE MOUSE (0) = 0: WEND: ms1 = MOUSE (1): ms2 = MOUSE (2)
IF ms1 > x1 AND ms1 < x2 AND ms2 > 36 AND ms2 < 50 THEN
risposta = 1: LINE (x1, 36) - (x2, 50), 1, bf: LOCATE 6, 1
PRINT PTAB (x1 + 8); scelta1$;: GOTO fine
END IF
IF ms1 > x3 AND ms1 < x4 AND ms2 > 36 AND ms2 < 50 THEN
risposta = 0: LINE (x3, 36) - (x4, 50), 1, bf: LOCATE 6, 1
PRINT PTAB (x3 + 8); scelta2$;: GOTO fine
END IF
WEND
fine:
WINDOW CLOSE 5
END SUB
```

pulsante sinistro mentre il puntatore si trova all'interno di riquadri rappresentanti le scelte.

Tutta la difficoltà, in fondo, sta unicamente nel calcolare esattamente dove posizionare le varie stringhe da inserire nel requester.

Un esempio pratico lo si può esaminare nel breve listato di queste pagine, che contiene una subroutine (**Sub...End Sub**) di nome **Requester**, resa il più autonoma possibile per facilitarne l'inserimento in qualunque listato.

La presentazione grafica, essenziale nonostante la praticità, può naturalmente essere modificata in accordo con i propri gusti.

La funzione Requester non adopera alcuna libreria di sistema.

Per utilizzarla (come mostrato nel banale demo del listato), occorre richiamarla inviando come parametri:

* **Due stringhe** che verranno visualizzate centrate nella parte superiore del requester (nel listato: **riga1$** e **riga2$**). La loro lunghezza massima è rapportata alle dimensioni della finestra-requester. Si noti l'uso della funzione **Window(2)**, che restituisce la larghezza della finestra corrente (nel nostro caso, il requester), espressa in **pixel**.

Quest'ultimo dato viene diviso per otto per determinare la lunghezza massima delle stringhe, in quanto ogni carattere (normalmente) occupa 8 pixel.

Qualora una stringa inviata fosse più lunga del consentito, essa verà semplicemente troncata dall'istruzione:

Left$ (rigax$, largmax)

Inutile aggiungere che, volendo, può essere modificata la dimensione di **Window 5** in accordo con le proprie esigenze.

La scelta di 5 come numero di finestra, è del tutto facoltativa, adottata più che altro per evitare conflitti con eventuali altre finestre aperte in precedenza dal programma chiamante.

* **Due stringhe** che rappresenteranno le due possibili scelte (nel listato: **scelta1$** e **scelta2$**). Queste, sempre in base alla loro lunghezza, verranno visualizzate in due riquadri colorati in nero: cliccando al loro interno si modificherà il valore del parametro descritto di seguito, permettendo la cosiddetta scelta "binaria". Al momento della scelta a suon di mouse, il "gadget" si illuminerà per un istante prima di restituire il controllo al programma principale. Nel demo, quest'ultimo si limita a stampare "SI" o "NO" a seconda della scelta effettuata, ma chiaramente si può dirottare il flusso generale verso le applicazioni che interessano.

* Una **variabile** numerica (nel listato: **risposta**) che conterrà, al "rientro" dal sottoprogramma, un valore **0** oppure **1** a seconda della scelta effettuata. Lo zero, in particolare, indicherà che è stato selezionato il riquadro di sinistra, mentre 1 indicherà la selezione del riquadro di destra.

In base a questo responso, è sufficiente una banale condizione If...Then...Else nella sezione principale del programma per verificare la scelta effettuata, ed agire di conseguenza. Sulla base di quanto illustrato, si disporrà in definitiva di una nuova funzione Basic adoperabile con la sintassi...

Requester stringa1$, stringa2$, scelta1$, scelta2$, risposta

...o, in alternativa, con...

Call Requester (stringa1$, stringa2$, scelta1$, scelta2$, risposta)

Il metodo evita le librerie di sistema, ma per migliorare le prestazioni di qualunque programma, prima o poi sarà necessario farvi ricorso.

by Bellussi Gabriele & Fiorucci Fabrizio.

ARRIVA MISTER X

Quando l'incognita si nasconde in una formula matematica, non sempre è agevole "estrarla". Due quindicenni provano a risolvere il problema, accettando la "sfida" proposta tempo addietro

Siamo due ragazzi quindicenni come tutti gli altri (tranne quando ci sediamo davanti al nostro **Amiga**...). Sul N. **66** di Commodore Computer Club veniva lanciata una sfida ai lettori nella quale **si chiedeva la realizzazione di un programma che fosse in grado (partendo da una qualsiasi equazione matematica) di estrarre tutte le formule inverse**. Noi ci siamo riusciti, anche se per raggiungere tale risultato abbiamo dovuto impiegare un anno della nostra miserevole vita...

Il programma è molto semplice da usare perchè è stata rispettata pienamente (o quasi) la sintassi propria della matematica e in più non presenta limitazioni rilevanti. L'unica pecca è l'impossibilità di calcolare il valore dell'incognita qualora essa si trovi come esponente di una potenza.

Le uniche variazioni per le quali la sintassi accettata dal programma differisce dalla sintassi matematica, sono rappresentate dal modo di esprimere funzioni quali il **coseno**, il **seno**, la **radice quadrata**, l'**esponente** di una potenza, ecc. (ma di questo riparleremo in seguito).

Altra caratteristica del programma proposto (che da adesso in poi chiameremo **X-Discoverer**) è la possibilità di aumentarne la potenza agendo su poche righe (ma anche di questo riparleremo in seguito).

Nei paragrafi successivi analizzeremo anche l'algoritmo utilizzato da X-Discoverer, ma già adesso anticipiamo che si tratta di un algoritmo molto semplice e molto simile al metodo utilizzato manualmente per estrarre la famigerata incognita.

Istruzioni per l'uso

Come precedentemente affermato, X-Discoverer è semplice da usare ma, nonostante ciò, vi sono alcune piccole regole alle quali attenersi per utilizzarlo:

1) La formula, ovviamente, deve essere digitata in forma lineare. Ciò significa che eventuali linee di frazione esigono l'uso di parentesi, come ad esempio...

a/b+c=d

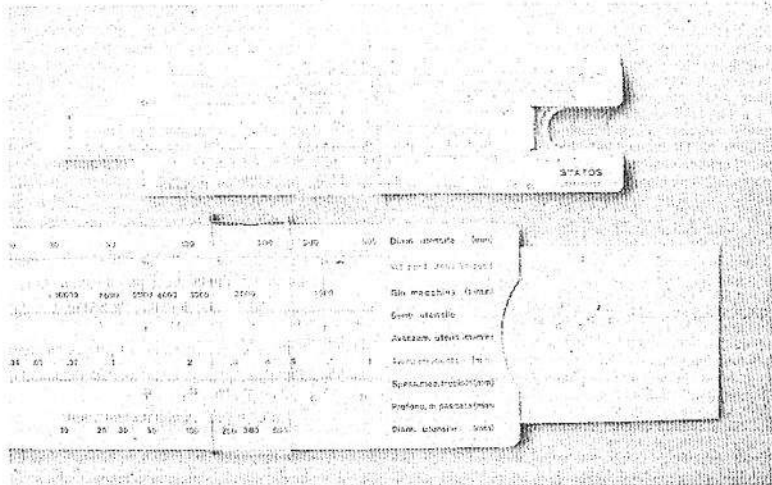
2) X-Discoverer interpreta il carattere **x** (minuscolo) come **incognita** dell'equazione; pertanto estrae la formula inversa in base a tale lettera e in base alla sua collocazione nella formula.

Per chiarire quanto detto, consideriamo la semplice formula **a * b = c** e supponiamo di voler ottenere la formula inversa per calcolare **b** noti **a** e **c.** Per ottenere la formula inversa, bisognerà sostituire la lettera **b** dell'equazione con la lettera **x** minuscola e dare in pasto a X-Discoverer l'equazione così ottenuta. In pratica, dopo il Run dovrete digitare...

a * x = c

...per avere, come output:

x = (c / a)

3) E' possibile sottintendere l'operazione di **moltiplicazione** (ad esempio **a * b = c** potrà essere espresso come **a b = c**). In caso di moltiplicazione o divisione tra un valore positivo e uno negativo, nella matematica è valida la sintassi a/**(-b)=c**, mentre X-Discoverer accetta anche **a/-b=c.**

```
' (AmigaBasic) X Discoverer
' by Gabriele Bellussi & Fabrizio Fiorucci 1991
main:
WIDTH 76
DIM vs$(100,1),p1%(20),p2%(20)
LOCATE 13,1
INPUT "formula :";f$
IF INSTR(f$,"x")<INSTR(f$,"=") THEN scompone
a$=MID$(f$,INSTR(f$,"=")+1,LEN(f$)-INSTR(f$,"=")+1)+"="
f$=a$+MID$(f$,1,INSTR(f$,"=")-1)
'------ scompone la formula nei suoi singoli elementi
scompone:
c=1:FOR i=1 TO LEN(f$):b$=MID$(f$,i,1)
IFb$="="THENc=c+1:ug=c:vs$(c,0)=b$:c=c+1:i=i+1:b$=MID$(f$,i,1)
b1$=RIGHT$(vs$(c,0),1):a%=(b1$>="a" AND b1$<="z")OR(b1$>="A" AND
b1$<="Z")
a%=a% OR(b1$>="0" AND b1$<="9") OR b1$=")" OR b1$="]" OR b1$=">"
IF flag THEN ciclo.semplice:ELSE IF a%=0 THEN aggiungi
IF b$="^" THEN flag=1:b2$="":GOTO fine.scomp
IF b$="]" OR b$="." THEN b2$="":GOTO fine.scomp
c=c+1:IF  b$="["  THEN  b2$="*+":b$=MID$(f$,i,5):i=i+4:GOTO  fi-
ne.scomp
IF INSTR("+-*/",b$) OR b$=")" THEN b2$="":GOTO fine.scomp
IF b$<"0" OR b$>"9" THEN b2$="*+":GOTO fine.scomp
IF b1$>="0" AND b1$<="9" THEN c=c-1:b2$="":ELSE:b2$="*+"
GOTO fine.scomp
aggiungi:
IF b1$="(" THEN c=c+1
IF  b1$="."  OR  b1$="+"  OR  b1$="-"  OR  b$="+"  OR  b$="-"  THEN
b2$="":ELSE:b2$="+"
IF b$="[" THEN b$=MID$(f$,i,5):i=i+4
GOTO fine.scomp
ciclo.semplice:
b2$="":IF b$=">" THEN flag=0
fine.scomp:
vs$(c,0)=vs$(c,0)+b2$+b$:IF INSTR(vs$(c,0),"x") THEN in=c
NEXT i
'---------- Estrae i coefficienti dell'incognita dalla formula
a=-1:fp=0:fs=0:c$=")":FOR i=in+1 TO ug-1                   :' ciclo
avanti
b$=LEFT$(vs$(i,0),1):lm=0:GOSUB sub.comune:NEXT
a=1:fp=0:fs=0:c$="(":FOR i=in-1 TO 0 STEP-1                :' ciclo
indietro
IF i=>0 THEN b$=RIGHT$(vs$(i,0),1):lm=1:GOSUB sub.comune
NEXT:GOTO elaborazione
sub.comune:
a$=LEFT$(vs$(i+lm,0),1):lm=0:np=0
IF a$="*" OR a$="/" THEN fs=1
IF a$="-" OR a$="+" THEN fp=1
IF b$=c$ THEN fs=0:fp=0:RETURN
IF fs AND fp THEN fp=0:fs=0:lm=-1
ciclo.parentesi:
IF LEFT$(vs$(i,0),1)=")" THEN np=np+a
IF RIGHT$(vs$(i,0),1)="(" THEN np=np-a
IF fs AND lm=0 THEN vs$(i,1)=vs$(i,0)
IF np>lm THEN i=i-a:IF i=>0 AND i<=ug-1 THEN ciclo.parentesi
RETURN
'---- aggiunge segni "*" ai coeff. che non l'hanno (ma con qualche
eccezione)
elaborazione:
np=0:FOR j=0 TO ug-1:IF np THEN fine.s
IF vs$(j,1)="" OR LEFT$(vs$(j,1),1)="*" OR LEFT$(vs$(j,1),1)="/"
THEN fine.s
fp=np:np=0:FOR k=j TO ug-1:a$=LEFT$(vs$(k,0),1):r=0:i=k
IF a$<>"/" AND np THEN pippo.segni
```

4) Le funzioni matematiche e le potenze devono essere espresse usando una sintassi leggermente differente da quella impiegata normalmente in matematica. La formula **a + cos (x) = b** dovrà essere espressa nel seguente modo:

a + [cos (x)] = b

Dall'esempio si nota che la funzione matematica è stata racchiusa all'interno di due **parentesi quadre**, per permettere al programma di **non** interpretarla come una serie di coefficienti di x (X-Discoverer considererebbe **a+cos(x)=b** come **a + c * o * s * (x) = b**.

L'esponente di una potenza deve essere preceduto dal simbolo "freccia in alto" (**^**) e racchiuso tra i segni "maggiore" e "minore" (**< >**).

Come esempio, consideriamo la formula...

a + x ^ 2 = b

...che andrà espressa come...

a + x ^ < 2 > = b

Si è resa necessaria tale sintassi perchè X- Discoverer accetta all'esponente anche espressioni, come ...

a + x ^ <b + c> = d

...e non solo valori semplici. Provando a far elaborare da X-Discoverer la formula dell'esempio precedente si otterrebbe, come output, la seguente formula:

x = (d - a) ^ < 1 / (b + c) >

Per chi non lo sapesse, **x^<1/2>** equivale alla radice quadrata di **x**.

5) Tra i simboli accettati da X-Discoverer figurano le lettere minuscole, le lettere maiuscole (in questo caso la X maiuscola non viene interpretata come incognita dell'equazione), costanti numeriche intere e/o decimali, i simboli **+**, **-**, *****, **/** e le parentesi tonde oltre al simbolo **=**.

E' molto importante ricordarsi di chiudere tutte le parentesi precedentemente aperte, controllare la presenza dell'incognita e dell'uguale all'interno dell'equazione, altrimenti X-Discoverer potrebbe bloccarsi emettendo una segnalazione di errore.

6) Se la formula digitata contiene funzioni matematiche, come ad esempio...

[cos (x)] = a

...X-Discoverer, una volta elaborata la formula, stamperà il seguente risultato:

x = ([a cos (a)])

...dove **[acos()]** è l'inverso di **[cos()]**, così come **[asin()]** è l'inverso di **[sin()]** etc.

```
ciclo:
GOSUB aggiorna.np:IF np THEN i=i+1:GOTO ciclo
IF in<=i THEN fs=0:k-ug-1
pippo.segni:
IF a$-"*" AND np=0 THEN fs=1
r=0:i=k:GOSUB   aggiorna.np:NEXT   k:np=fp:IF   fs   THEN
vs$(j,1)="*"+vs$(j,1)
fine.s:
r=1:i-j:GOSUB aggiorna.np:NEXT j
' memorizza in pl% e in p2% i puntatori ai livelli di tonde
j=1:np-0:r=0:pl%(0)=in:p2%(0)=in:vs$(ug,0)=""
FOR i=in+1 TO ug-1:GOSUB aggiorna.np
IF np<0 THEN p2%(j)=i:np-0:j=j+1
NEXT i:j=1:np=0
FOR i-in TO 0 STEP -1:GOSUB aggiorna.np
IF np>0 THEN pl%(j)=i:np=0:j=j+1
NEXT i:pl%(j)=0:p2%(j)=ug
'---------- sposta il membro non contenente la x in fi$
FOR i=ug+1 TO c:fi$=fi$+vs$(i,0):NEXT
GOSUB scanner  :'------ vai a scanner
'---------- inverte i segni + e - ai non coeff. di x
FOR k=j TO 1 STEP -1:r=0:np=0:fl-0:lm=0  :'loop principale
' primo loop secondario: cambia segno ai non coefficienti
np=0:FOR i=pl%(k)+1 TO p2%(k)-1:IF vs$(i,1)<>"" THEN fine.segni
IF i=pl%(k-1) THEN i=p2%(k-1):GOTO fine.segni
IF  LEFT$(vs$(i,0),1)="+"  AND  np=0  THEN  MID$(vs$(i,0),1,1)-"-
":fl=1:GOTO ag.s
IF LEFT$ (vs$(i,0),1)="-"ANDnp=0THEN MID$(vs$(i,0),1,1)="+":fl=1
ag.s:
GOSUB aggiorna.np:fi$=fi$+vs$(i,0)
fine.segni:
NEXT i:IF fl THEN fi$="("+fi$+")"
'---------- secondo loop secondario (che bel gioco di parole!)
' sposta i coefficienti divisi da x, ad esempio : a/x=c ---> x=a/c
IF LEFT$(vs$(pl%(k-1),0),1)<>"/" THEN scambia.coefficienti
lm=1:fi$="/("+fi$+")"
FOR i=pl%(k-1) TO pl%(k)+1 STEP -1:fi$=vs$(i,1)+fi$
vs$(i,1)="":NEXT i
scambia.coefficienti:
'---------- terzo loop secondario
' scambia i segni * e / ai coeff. di x e li pone in coda a fi$
fl-0:r=1:np=0:FOR i=pl%(k)+1 TO p2%(k)-1
IF i=pl%(k-1) THEN i=p2%(k-1)+1
IF NOT(lm=0 OR i=>0) THEN fine.coefficienti
IF      LEFT$(vs$(i,1),1)="*"      AND      np=0      THEN
MID$(vs$(i,1),1,1)="/":fl=1:GOTO agg
IF      LEFT$(vs$(i,1),1)="/"      AND      np=0      THEN
MID$(vs$(i,1),1,1)="*":fl=1
agg:
GOSUB aggiorna.np
fine.coefficienti:
fi$=fi$+vs$(i,1):vs$(i,1)=""
NEXT i
IF fl THEN fi$="("+fi$+")"
lm=0:GOSUB pot.fun:NEXT k
IF MID$(vs$(in,0),INSTR(vs$(in,0),"x")-1,1)="-" THEN fi$-"-"+fi$
PRINT:PRINT"La formula inversa e':":PRINT:PRINT"x-";fi$:END
'-------------------- subroutines ----------------------
aggiorna.np:
IF RIGHT$(vs$(i,r),1)="(" THEN np=np+1
IF LEFT$(vs$(i,r),1)=")" THEN np=np-1
RETURN
pot.fun:
IF INSTR(vs$(p2%(k-1),0),">")=0 THEN funzioni
```

L'algoritmo

Ma come diavolo riesce X-Discoverer ad estrarre questa benedetta incognita?

Prima di rispondere, sarebbe meglio ripassare un po' di algebra utilizzando una formula di esempio:

a + (b - c) / (d - x) f = e

Se provassimo a trovare manualmente la formula inversa per calcolare **x** dovremmo, prima di tutto , trasportare **+a** al membro non contenente x, invertendo il segno (**+a** diventerà **-a**); successivamente dovremo dividere entrambi i membri per il coefficiente **f**, quindi moltiplicarli per **(d-x)**. A questo punto non resterà altro da fare che dividere entrambi i membri per **((e-a)/f)**, per poi trasportare **d** al membro non contenente **x**, invertendone il segno. Ma non finisce qui, perchè **x** è preceduta dal segno negativo; quindi, per eliminarlo, basterà sostituire il segno ad x e racchiudere l'altro membro in un livello di parentesi preceduto dal segno negativo. Il procedimento descritto è simile a quello adottato da X-Discoverer, che permette una affidabilità pressochè totale.

Adesso vedremo come l'algoritmo impiegato dal programma riesce ad estrarre la formula inversa, partendo dall'esempio precedente.

Prima di tutto la stringa-formula immessa dall'utente viene suddivisa nei singoli elementi che la compongono e memorizzata nella riga **0** della matrice **VS$**. Vediamo, con un esempio, come avviene la scomposizione della formula.

La stringa - formula...

a + (b - c) / (d - x) f = e

...diventa...

+a + (+b - c) /+ (+d - x) * + f = + e

Dall'esempio si nota che i segni algebrici + e i segni di moltiplicazione vengono espressi esplicitamente (mentre di solito sono sottointesi). La scomposizione serve per facilitare le fasi successive, che altrimenti risulterebbero eccessivamente lunghe e complesse. Una volta tritata la stringa-formula e memorizzata in **VS$**, vengono estratti i coefficienti dell'incognita e memorizzati nella riga 1 della suddetta matrice.

In questo caso i coefficienti saranno:

VS$ (n, 0):
+a + (+b - c) /+ (+d - x) * + f = + e

VS$ (n, 1):
+ (+b - c) * + f

```
w=INSTR(vs$(p2%(k-1),0),"<")+1
a$=MID$(vs$(p2%(k-1),0),w,LEN(vs$(p2%(k-1),0))-w)
fi$=fi$+"^<1/("+a$+")>"
funzioni:
IF LEFT$(RIGHT$(vs$(p1%(k-1),0),5),1)<>"[".THEN RETURN
fi$="[a"+LEFT$(RIGHT$(vs$(p1%(k-1),0),4),3)+"("+fi$+")]":RETURN
scanner:
r=0:FOR k=j-1 TO 1 STEP -1:np=0
IF LEFT$(vs$(p1%(k),0),1)<>"-" AND MID$(vs$(p1%(k),0),2,1)<>"-"
THEN endscan
FOR i=p1%(k)+1 TO p2%(k)-1:IF np THEN aggiorna.scan
a$=LEFT$(vs$(i,0),1):IF a$="-" THEN a$="+":GOTO inverti
IF a$="+" THEN a$="-":ELSE GOTO aggiorna.scan
inverti:
MID$(vs$(i,0),1,1)=a$:IF vs$(i,1)<>"" THEN MID$(vs$(i,1),2,1)=a$
aggiorna.scan:
GOSUB aggiorna.np:NEXT i
endscan:
NEXT k:RETURN
```

I coefficienti vengono infatti memorizzati in riga 1 nella medesima posizione **n** che occupavano in riga 0. Successivamente vengono memorizzati in due vettori numerici interi (**p1%** e **p2%**) i puntatori ai livelli di parentesi, tenendo conto che per **livello di parentesi** si intende una coppia di parentesi tonde aperta - chiusa contenenti l'incognita; più semplicemente...

```
+a +( +b -c ) /+( +d -x ) *+f = +e
2               1       01     2
```

...in cui le freccette (^) indicano i livelli di parentesi). Il vettore **P1%(n)** conterrà i puntatori alle tonde aperte dei livelli, mentre il vettore **P2%(n)** conterrà i puntatori alle rispettive tonde chiuse.

I livelli di parentesi **0** e **2** vengono assunti per default e puntano, rispettivamente, all'incognita e agli estremi del membro contenente l'incognita x.

Le operazioni precedentemente descritte hanno lo scopo di trattare la formula in modo tale da poterla elaborare con facilità perchè se, ad esempio, non ci fosse la parte di programma adibita alla scomposizione della formula, si dovrebbero svolgere continue operazioni di taglia e incolla, che allungherebbero notevolmente il programma.

Detto questo, continuiamo la nostra esplorazione parlando, finalmente, dell'elaborazione vera e propria della formula. La prima operazione svolta consiste in una **scansione** completa della formula, al fine di controllare la presenza di livelli di parentesi negativi e agendo di conseguenza (tale scansione viene svolto dalla subroutine **scanner**).

Successivamente il membro non contenente la x viene memorizzato nella variabile **FI$** (destinata a contenere la formula inversa) e cancellato dalla matrice **VS$(n,0)**. Fatto ciò, si entra nell'elaborazione vera e propria costituita da un ciclo principale (For K = J - 1 To Step - 1 ..... next K) nel quale sono contenute tre sezioni di programma costituite da altrettanti cicli secondari con compiti ben distinti l'uno dall'altro. Il ciclo principale ha lo scopo di far puntare i cicli secondari ai vari livelli di parentesi presenti nella formula, partendo dal livello più esterno per arrivare a quello più interno. Nel caso dell'esempio precedente, il ciclo principale partirà dal livello numero 2 e terminerà al livello numero 0.

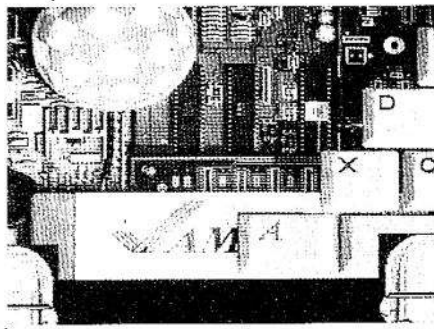
I cicli secondari sono tre: il **primo** serve a sostituire i segni algebrici ai non coefficienti presenti nel livello puntato dal ciclo principale e ad appenderli in coda a **FI$**, il **secondo** serve per verificare l'esistenza di un coefficiente diviso da x (nel caso dell'esempio: il coefficiente **(b-c)**) e ad "appenderlo" in testa a **FI$**; il **terzo** serve per invertire i segni "asterisco" e "diviso" (* /) ai coefficienti di x e ad appenderli in coda a **FI$**.

E' importantissimo notare che il primo e il terzo ciclo secondario elaborano gli elementi presenti nel livello puntato da P1%(k) e da P2%(k) ignorando i livelli inferiori; infatti in questi due cicli è presente la riga **If I= P1%(K-1) Then I= P2% (K-1)+1** che permette di "saltare" i livelli di parentesi inferiori a quello attuale. Oltre ai tre loop secondari, sono presenti due subroutines che intervengono nell'elaborazione della formula: scanner (già descritta precedentemente) e **pot-fun**.

La subroutine pot-fun è divisa in due parti: la prima entra in azione nel caso in cui il livello di parentesi puntato da P1%(k) e da P2%(k) sia dotato di esponente, la seconda parte entra in azione nel caso in cui il suddetto livello di parentesi sia all'interno di una funzione matematica. Una volta terminato il ciclo principale, il programma verifica se l'incognita è preceduta dal segno algebrico negativo e, in caso affermativo, aggiunge il segno meno in testa a **FI$**.

Dall'analisi del programma si nota come X-Discoverer non sia altro che un elaboratore di stringhe un po' particolare in quanto la procedura seguita non prevede alcun tipo di calcolo matematico.

Modifiche

X-Discoverer accetta un massimo di **venti livelli di parentesi** (tanti quanti sono quelli accettati da una comune calcolatrice scientifica di buona qualità) ma è possibile aumentarne il numero semplicemente modificando la capacità dei vettori P1% e P2%, ricordandosi di fare in modo che i due vettori siano uguali. Modificando l'ampiezza di **VS$** sarà possibile inserire formule più lunghe (ma riteniamo che non sia necessaria una modifica del genere). Per i più volenterosi, proponiamo anche l'eliminazione della pecca citata nell'introduzione, oppure l'inserimento di un'opzione tramite la quale calcolare il valore numerico di x noto il valore degli altri elementi

QUANTO COSTA IL TUO COMMODORE

Amiga 2000 - L. 2.715.000

Microprocessore Motorola MC68000 - Clock 7.16MHz - Kickstart ROM - Memoria RAM: 1 MByte - 3 chip custom per DMA, Video, Audio, I/O - 5 Slot di Espansione Amiga Bus 100 pin AutoconfigTM - 1 Slot di Espansione 86 pin per Schede Coprocessore - 2 Slot di Espansione compatibili AT/XT - 2 Slot di Espansione compatibili XT - 2 Slot di Espansione Video - 1 Floppy Disk Drive da 3 1/2", 880 KBytes - Porta seriale RS232C - Sistema Operativo single-user, multitasking AmigaDOS - Compatibilità MS-DOS XT/AT disponibile con schede interne Janus (A2088 - A2286) - Monitor escluso

Amiga 500 - L. 995.000

Microprocessore Motorola MC68000 - Clock 7.16 MHz - Kickstart ROM - Memoria RAM: 512 KBytes - 3 Chip custom per DMA, Video, Audio, I/O - 1 Floppy Disk Driver da 3 1/2", 880 KBytes - Porta seriale RS232C - Porta parallela Centronics

Videomaster 2995 - L. 1.200.000

Desk Top Video - Sistema per elaborazini video semiprofessionale composto da genlock, digitalizzatore e alloggiamento per 3 drive A2010 - Ingressi videocomposito (2), RGB - Uscite Videocomposito, RF, RGB + sync -

Floppy Disk Driver A 1010 - L. 335.000

Floppy Disk Driver - Drive esterno da 3 1/2" - Capacità 880 KBytes - Collegabile a tutti i modelli della linea Amiga, alla scheda A2088 e al PC1

Floppy Disk Drive A 2010 - L. 280.000

Floppy Disk Drive - Drive interno aggiuntivo da 3 1/2" - Capacità 880 KBytes - Collegabile ad Amiga 2000

Hard Disk A 590 - L. 1.750.000

Hard Disk+Controller+RAM - Scheda Controller - Hard Disk da 3 1/2" 20 MBytes - 2 MBytes "fast" RAM - Collegabile all'Amiga 500

Scheda Janus A 2088 + A 2020 - L. 1.050.000

Scheda Janus XT+Floppy Disk Drive da 5 1/4", 360 KBytes - Scheda Bridgeboard per compatibilità MS-DOS (XT) in Amiga 2000 - Microprocessore Intel 8088 - Coprocessore matematico opzionale Intel 8087

A2286+A2020 - L. 1.985.000

Scheda Janus AT+Floppy Disk Drive da 5 1/4", 1.2 MBytes - Scheda Bridgeboard per compatibilità MS-DOS (AT) in Amiga 2000 - Microprocessore Intel 80287 - Clock 8 MHz - RAM: 1 MBytes on-board - Floppy Disk Controller on-board - Floppy Disk Driver disegnato per l'installazione all'interno dell'Amiga 2000 -

Scheda A2620 - L. 2.700.000

Scheda Processore Alternativo 32 bit - Scheda per 68020 e Unix - Microprocessore Motorola MC68020 - Coprocessore matematico Motorola MC68881 (opzionale MC68882)

Scheda A Unix - L. 3.250.000

Sistema Operativo AT&T Unix System V Release 3 - Per Amiga 2000 con scheda A2620 e Hard Disk 100 MBytes

Hard Disk A2092+PC5060 - L. 1.020.000

Hard Disk e controller - Hard Disk 3 1/2" ST506 - Capacità formattata 20 MBytes

Hard Disk A2090+2092 - L. 1.240.000

Hard Disk e controller - Hard Disk 3 1/2" ST506 - Capacità formattata 20 MBytes

Hard Disk A2090+A2094 - L. 1.900.000

Stesse caratteristiche del kit A2092 ma con disco da 40 MBytes

Espansione di memoria A2058 - L. 1.149.000

Espansione di memoria - Scheda di espansione per Amiga 2000 - Fornita con 2 MBytes "fast" RAM, espandibile a 4 o 8 MBytes

Scheda Video A2060 - L. 165.000

Modulatore video - Scheda modulatore video interna per Amiga 2000 - Uscite colore e monocromatica - Si inserisce nello slot video dell'Amiga 2000

Genlock Card A2301 - L. 420.000

Genlock - Scheda Genlock semiprofessionale per Amiga 2000 - Permette di miscelare immagini provenienti da una sorgente esterna con immagini provenienti dal computer

Professional Video Adapter Card A2351 - L. 1.500.000

Professional Video Adapter - Scheda Video Professionale per Amiga 2000 (B) - Genlock qualità Broadcast - Frame Grabber - Digitalizzatore - Include software di controllo per la gestione interattiva
(Disponibile da maggio '89)

A501 - L. 300.000

Espansione di memoria - Cartuccia di espansione di memoria da 512 KBytes per A500

A520 - L. 45.000

Modulatore RF - Modulatore esterno A500 - Permette di connettere qualsiasi televisore B/N o colori ad Amiga 500

A Scart - L. 27.000

Cavo di collegamento A500/A2000 con connettore per televisione SCART

Monitor a colori 1084 - L. 595.000

Monitor a colori ad alta risoluzione - Tubo 14" Black Matrix antiriflesso - Pitch 0.39 mm - Compatibile con Amiga 500/2000, PC (tutta la gamma), C64 e C128

Monitor a colori 2080 - L. 770.000

Monitor a colori ad alta risoluzione e lunga persistenza - Tubo 14" Black Matrix antiriflesso - Pitch 0.39 mm - Frequenza di raster 50 Hz - Compatibile con Amiga 500/2000, PC (tutta la gamma), C64 e C128

Monitor Monocromatico A2024 - L. 1.235.000

Monitor monocromatico a fosfori "bianco-carta" - Turbo 14" antiriflesso - (Disponibile da marzo '89)

PC60/40 - L. 7.812.000

Microprocessore Intel 80386 - Coprocessore matematico opzionale Intel 80387 - Clock 8 o 16 MHz selezionabile via software e da tastiera - Monitor monocromatico 14" - Tastiera avanzata 102 tasti con 12 funzioni - Sistema Operativo MS-DOS 3 2.1 - Interprete GW-Basic

PC60/40C - L. 8.127.000

Stessa configurazione ma con monitor 14" a colori mod. 1084

PC 60/80 - L. 10.450.000

Microprocessore Intel 80386 - Coprocessore opzionale Intel 80387 - Clock 8 o 16 MHz selezionabile via software e da tastiera - Memoria RAM: 2.5 MBytes - 1 Floppy Disk Drive da 5 1/4", 1.2 MBytes - 1 Floppy Disk Drive opzionale da 3 1/2", 1.44 MBytes - 1 Hard Disk da 80 MBytes - 2 Porte parallele Centronics - Mouse video EGA (compatibile MDA - Hercules - CGA). Emulazioni disponibili via hardware e software - Monitor monocromatico 14" - Tastiera avanzata 102 tasti con 12 tasti funzione - Sistema Operativo MS-DOS 3.21 - Ambiente Operativo Microsoft Windows/386 - Interprete GW-Basic

PC60/80C - L. 10.700.000

Stessa configurazione ma con monitor 14" a colori mod. 1084

PC40/20 - L. 4.100.000

Microprocessore Intel 80286 - Coprocessore matematico opzionale Intel 80287 - Clock 6 o 10 MHz selezionabile via software, hardware o da tastiera - Memoria RAM: 1 MByte - 1 Floppy Disk Drive da 5 1/4", 1.2 MBytes - 1 Hard Disk da 20 MBytes - Porta seriale RS232 - Porta parallela Centronics - Scheda video AGA multistandard (MDA - Hercules - CGA) Emulazioni disponibili via hardware e software - Monitor monocromatico 14" - Tastiera avanzata 102 tasti con 12 tasti funzione - Sistema Operativo MS-DOS 3.21 - Interprete GW-Basic

PC40/20C - L. 4.350.000

Stessa configurazione ma con monitor 14" a colori mod. 1084

PC 40/40 - L. 5.285.000

Microprocessore Intel 80286 - Coprocessore matematico opzionale Intel 80287 - Clock 6 o 10 MHz selezionabile via software, hardware o da tastiera - Memoria RAM: 1 MByte - 1 Floppy Disk Drive da 5 1/4", 1.2 MBytes - 1 Hard Disk da 20 MBytes - Porta seriale RS232 - Porta parallela Centronics - Scheda video AGA multistandard (MDA - Hercules - CGA) Emulazioni disponibili via hardware e software - Monitor monocromatico 14" - Tastiera avanzata 102 tasti con 12 tasti funzione - Sistema Operativo MS-DOS 3.21 - Interprete GW-Basic

PC40/40C - L. 5.535.000

Stessa configurazione ma con monitor 14" a colori mod. 1084

1352 - L. 78.000

Mouse - Collegabile con Microsoft Bus Mouse - Collegabile direttamente a PC1, PC10/20 - III, PC40 - III

PC910 - L. 355.000

Floppy Disk Drive - Drive interno aggiuntivo da 3 1/2" per PC10/20-I-II-III - Capacità 360 o 720 KBytes selezionabile tramite "config. sys" - Corredo di telaio di supporto per l'installazione in un alloggiamento per un drive da 5 1/4" - Interfaccia identica ai modelli da 5 1/4"

PC1 - L. 995.000

Microprocessore Intel 8088 - 1 Floppy Disk Drive da 5 1/4" - Porta seriale RS232C - Porta parallela Centronics - - Monitor monocromatico 12" - Tastiera 84 tasti - Sistema Operativo MS-DOS 3.2 - Interprete GW-Basic

PCEXP1 - L. 640.000

PC Expansion Box - Box esterno di espansione per PC 1 - Alimentatore aggiuntivo incluso - Contiene 3 Slot di Espansione compatibili Ibm XT - Alloggiamento per Hard Disk da 5 1/4" - Si posiziona sotto il corpo del PC1 e viene collegato tramite degli appositi connettori

PC10-III - L. 1.360.000

Microprocessore Intel 8088 Clock 4.77 MHz 9.54 MHz (double) selezionabile via software e da tastiera - Memoria RAM: 640 KBytes - 2 Floppy Disk Drive da 5 1/4", 360 KBytes - Porta seriale RS232C - Porta parallela Centronics - Porta Mouse per Mouse Commodore 1352 (compatibile Microsoft Bus Mouse - Tastiera avanzata 102 con 12 tasti funzione Sistema Operativo MS-DOS 3.21 - Interprete GW-Basic

PC10-IIIC - L. 1.675.000

Stessa configurazione ma con monitor 14" a colori mod. 1084

PC20-III - L. 2.095.000

Microprocessore Intel 8088 - Clock 4.77 MHz 9.54 MHz (double) selezionabile via software e da tastiera - 1/4", 360 KBytes - 1 Hard Disk da 20 MBytes - Porta seriale RS232C - Porta parallela Centronics - Porta Mouse per Mouse Commodore 1352 (compatibile Microsoft Bus Mouse) - Tastiera avanzata 102 con 12 tasti funzione Sistema Operativo MS-DOS 3.21 - Interprete GW-Basic

PC20-IIIC - L. 2.410.000

Stessa configurazione ma con monitor 14" a colori mod. 1084

Nuovo C64 - L. 325.000

Nuovo Personal Computer CPU 64 KBytes RAM - Vastissima biblioteca software disponibile - Porta seriale Commodore - Porta registratore per cassette - Porta parallela programmabile -

C128D - L. 895.000

Personal Computer CPU 128 KBytes RAM espandibile a 512 KBytes - ROM 48 KBytes - Basic 7.0 - Tastiera separata - Funzionante in modo 128,64 o CP/M 3.0 - Include floppy disk drive da 340 KBytes

Floppy Disk Drive 1541 II - L. 365.000

Floppy Disk Drive - Floppy Disk Drive da 5 1/4" singola faccia - Capacità 170 KBytes - Alimentazione separata - Compatibile con C64, C128, C128D

Floppy Disk Dirve 1581 - L. 420.000

Floppy Disk Drive da 3 1/2" doppia faccia - Capacità 800 KBytes - Alimentazione separata - Compatibile con C64, C128, C128D

1530 - L. 55.000

Registratore a cassette per C64, C128, C128D

Accessori per C64 - 128D

1700 - Espansione di memoria - Cartuccia di espansione di memoria a 128 KBytes per C128 **- L. 170.000**

1750 - Espansione di memoria - Cartuccia di espansione di memoria 512 KBytes per C128 - **L. 245.000**

1764 - Espansione di memoria - Cartuccia di espansione di memoria a 256 KBytes per C64 - Fornita di alimentatore surdimensionato - **L. 198.000**

16499 - Adattatore Telematico Omologato - Collegabile al C64 - Permette il collegamento a Videotel, P.G.E. e banche dati **L. 149.000**

1399 - Joystick - Joystick a microswitch con autofire - **L. 29.000**

1351 - Mouse - Mouse per C64, C128, C128D - **L. 72.000**

Monitor Monocromatico 1402 - L. 280.000

Monitor monocromatico a fosfori "bianco-carta" - Turbo 12" antiriflesso - Ingresso TTL - Compatibile con tutta la gamma PC

Monitor Monocromatico 1404 - L. 365.000

Monitor monocromatico a fosfori ambra - Turbo 14" antiriflesso a schermo piatto - Ingresso TTL - Compatibile con tutta la gamma PC - Base orientabile

Monitor Monocromatico 1450 - L. 470.000

Monitor monocromatico BI-SYNC a fosfori "bianco-carta" - Turbo 14" antiriflesso - Ingresso analogico e digitale - Doppia frequenza di sincronismo orizzontale per compatibilità con adattatori video MDA, Hercules, CGA, EGA e VGA

Monitor a colori 1802 - L. 445.000

Monitor a colori - Turbo 14" - Collegabile a C64, C128, C128D

Monitor monocromatico 1900 - L. 199.000

Monitor monocromatico a fosfori verdi - Turbo 12" antiriflesso - Ingresso videocomposito - Compatibile con tutta la gamma Commodore

Monitor a colori 1950 - L. 1.280.000

Monitor a colori BI-SYNC alta risoluzione - Turbo 14" antiriflesso - Ingresso analogico e digitale - Doppia frequenza di sincronismo orizzontale per compatibilità con adattatori video MDA, Hercules, CGA, EGA e VGA

Stampante MPS 1230 - L. 465.000

Stampante a matrice di punti - Testina a 9 aghi - 120 cps - Bidirezionale - 80 colonne - Near Letter Quality - Stampa grafica - Fogli singoli e modulo continuo - Trascinamento a trattore e/o frizione - Interfaccia seriale Commodore e parallela Centronics - Compatibile con tutti i prodotti Commodore

MPS 1230R - L. 19.000

Nastro per stampante

Stampante MPS 1500C - L. 495.000

Stampante a colori a matrice di punti - Testina a 9 aghi - 130 cps - Bidirezionale - 80 colonne - Supporta nastro a colori o nero - Near Letter Quality - Stampa grafica - Fogli singoli e modulo continuo - Trascinamento a trattore e/o frizione - Interfaccia parallela Centronics - Compatibile con la gamma Amiga e PC

MPS1500R - L. 37.000

Nastro a colori per stampante

MPS1500R - L.37.000

Nastro a colori per stampante

Stampante MPS 1550C - L. 575.000

Stampante a colori a matrice di punti - Testina a 9 aghi - 130 cps - Bidirezionale - 80 colonne - Supporta nastro a colori o nero - Near Letter Quality - Stampa grafica - Fogli singoli e modulo continuo - Trascinamento a trattore e/o frizione - Interfaccia seriale Commodore e parallela Centronics - Compatibile con tutti i prodotti Commodore

I COMMODORE POINT

LOMBARDIA

Milano
• AL RISPARMIO - V.LE MONZA 204
• BCS - VIA MONTEGANI 11
• BRAHA A. - VIA PIER CAPPONI 5
• E.D.S. - C.SO PORTA TICINESE 4
• FAREF - VIA A. VOLTA 21
• FLOPPERIA - V.LE MONTENERO 31
• GBC - VIA CANTONI 7 - VIA PETRELLA 6
• GIGLIONI - V.LE LUIGI STURZO 45
• L'UFFICIO 2000 - VIA RIPAMONTI 213
• LOGITEK - VIA GOLGI 60
• LU - MEN - VIA SANTA MONICA 3
• MARCUCCI - VIA F.LLI BRONZETTI 37
• MELCHIONI - VIA P. COLLETTA 37
• MESSAGGERIE MUSICALI - GALLERIA DEL CORSO 2
• NEWEL - VIA MAC MAHON 75
• PANCOMMERZ ITALIA - VIA PADOVA 1
• SUPERGAMES - VIA VITRUVIO 38
• 68000 E DINTORNI - VIA WASHINGTON 91
Provincia di Milano
• GINO FERRARI CENTRO HI-FI - VIA MADRE CABRINI 44 - S. ANG. LODIGIANO
• F.LLI GALIMBERTI - VIA NAZIONALE DEI GIOVI 28/36 - BARLASSINA
• TECNOLUX - VIA PIETRO NENNI 5 - BERNATE TICINO
• OGGIONI & C. - VIA DANTE CESANA 27 - CARATE BRIANZA
• AL RISPARMIO - VIA U. GIORDANO 57 - CINISELLO BALSAMO
• GBC - V.LE MATTEOTTI 66 - CINISELLO BALSAMO
• CASA DELLA MUSICA - VIA INDIPENDENZA 21 - COLOGNO MONZESE
• PENATI - VIA VERDI 28/30 - CORBETTA
• EPM SYSTEM - V.LE ITALIA 12 - CORSICO
• P.G. OSTELLARI - VIA MILANO 300 - DESIO
• CENTRO COMPUTER PANDOLFI - VIA CORRIDONI 18 - LEGNANO
• COMPUTEAM - VIA VECELLIO 41 - LISSONE
• M.B.M. - C.SO ROMA 112 - LODI
• L'AMICO DEL COMPUTER - VIA CASTELLINI 27 - MELEGNANO
• BIT 84 - VIA ITALIA 4 - MONZA
• IL CURSORE - VIA CAMPO DEI FIORI 35 - NOVATE MIL.
• I.C.O. - VIA DEI TIGLI 14 - OPERA
• R & C ELGRA - VIA SAN MARTINO 13 - PALAZZOLO MIL.
• ESSEGIEMME SISTEMI SAS - VIA DE AMICIS 24 - RHO
• TECNO - CENTRO - VIA BARACCA 2 - SEREGNO
• NIWA HARD&SOFT - VIA B. BUOZZI 94 - SESTO SAN GIOV.
• COMPUTER SHOP - VIA CONFALONIERI 35 - VILLASANTA
• ACTE - VIA B. CREMIGNANI 13 - VIMERCATE
• IL COMPUTER SERVICE SHOP - VIA PADANA SUPERIORE 197 - VIMODRONE
Bergamo
•D.R.B. - VIA BORGO PALAZZO 65
• TINTORI ENRICO &C. - VIA BROSETA 1
• VIDEO IMMAGINE - VIA CARDUCCI c/o CITTA' DI MERCATO
Provincia di Bergamo
BERTULEZZI GIOVANNI - VIA FANTONI 48 - ALZANO LOMBARDO
• COMPUTER SHOP - VIA VITTORIO VENETO 9 - CAPRIATE SAN GERVASIO
• B M R - VIA BUTTARO 4/T - DALMINE
• MEGABYTE 2 - VIA ROMA 61/A - GRUMELLO
• OTTICO OPTOMETRISTA ROVETTA - P.ZZA GARIBALDI 6 - LOVERE
• COMPUTER POINT - VIA LANTIERI 52 - SARNICO
• A.B. INFORMATICA - STRADA STATALE CREMASCA 66 - URGNANO
Brescia
• MASTER INFORMATICA - VIA F.LLI UGONI 10/B
PROVINCIA DI BRESCIA
• MISTER BIT - VIA MAZZINI 70 - BRENO
• CAVALLI PIETRO - VIA 10 GIORNATE 14 BIS - CASTREZZATO
• VIETTI GIUSEPPE - VIA MILANO 1/B - CHIARI
• MEGABYTE - P.ZZA MALUEZZI 14 - DESENZANO DEL GARDA
• BARESI RINO &C. - VIA XX SETTEMBRE 7 - GHEDI
• INFO CAM - VIA PROVINCIALE 3 - GRATACASOLO
• "PAC-LAND" di GARDONI - CENTRO COM.le - LA CASA DI MARGHERITA D'ESTE - VIA GIORGIONI 21
Como
• IL COMPUTER - VIA INDIPENDENZA 90
• 2M ELETTRONICA - VIA SACCO 3
Provincia di Como
• ELTRON - VIA IV NOVEMBRE 1 - BARZANO'
• DATA FOUND - VIA A. VOLTA 4 -ERBA
• CIMA ELETTRONICA - VIA L. DA VINCI 7 - LECCO
• FUMAGALLI - VIA CAIROLI 48 -LECCO
• RIGHI ELETTRONICA - VIA G. LEOPARDI 26 - OLGIATE COMASCO
Cremona
• MONDO COMPUTER - VIA GIUSEPPINA 11/B
• PRISMA - VIA BUOSO DA DOVARA 8
• TELCO - P.ZZA MARCONI 2/A
Provincia di Cremona
• ELCOM - VIA IV NOVEMBRE 56/58 - CREMA
• EUROELETTRONICA - VIA XX SETTEMBRE 92/A - CREMA
Mantova
• COMPUTER CANOSSA - GAL. FERRI 7
• 32 BIT - VIA C. BATTISTI 14
• ELET. di BASSO - V.LE RISORGIMENTO 69
Provincia di Mantova
• CLICK - ON COMPUTER - S.S. GOITESE 168 - GOITO
Pavia
• POLIWARE - C.SO C. ALBERTO 76
• SENNA GIANFRANCO - VIA CALCHI 5
Provincia di Pavia
• A. FERRARI - C.SO CAVOUR 57 - MORTARA
• LOGICA MAINT - V.LE M.TE GRAPPA 32 - VIGEVANO
• M. VISENTIN - C.SO V. EMANUELE 76 - VIGEVANO
Sondrio
• CIPOLLA MAURO - VIA TREMOGGE 25
Provincia di Sondrio
• FOTONOVA - VIA VALERIANA 1 - S.PIETRO DI BERBENNO
Varese
• ELLE - EFFE - VIA GOLDONI 35
• IL C.TRO ELET. - VIA MORAZZONE 2
• SUPERGAMES - VIA CARROBBIO 13
Provincia di Varese
• BUSTO BIT - VIA GAVINANA 17-BUSTO A.
• MASTER PIX - VIA S.MICHELE 3 - BUSTO A.
• PUNTO UFFICIO - VIA R.SANZIO 8 - GALLARATE
• GRANDI MAGAZZINI BOSSI - VIA CLERICI 196 - GERENZANO
• J.A.C. - C.so MATTEOTTI 38 - SESTO C.

PIEMONTE

Alessandria
• BIT MICRO - VIA MAZZINI 102
• SERV. INFOR. - VIA ALESSANDRO III 47
Provincia di Alessandria
• SONY ITALIANA - VIA G. MANARA 7 - CASALE MONFERRATO
• SGE ELETTRONICA - VIA BANDELLO 19 - TORTONA
• COMPUTER TEMPLE - VIA F. CAVALLOTTI 13 - VALENZA
Asti
• ASTI GAMES - C.SO ALFIERI 26
• RECORD - C.SO ALFIERI 166/3 (Galleria Argenta)
Cuneo
• ROSSI COMPUTERS - C.SO NIZZA 42
Provincia di Cuneo
• PUNTO BIT - C.SO LANGHE 26/C - ALBA
• BOSETTI - VIA ROMA 149 - FOSSANO
• COMPUTERLAND - VIA MAZZINI 30/32 - SALUZZO
Novara
• PROGRAMMA 3 - V.LE BUONARROTI 8
• PUNTO VIDEO - C.so RISORGIMENTO 39/B
Provincia di Novara
• COMPUTER - VIA MONTE ZEDA 4 -ARONA
• ALL COMPUTER - C.SO GARIBALDI 106 - BORGOMANERO
• S.P.A. - C.SO DISSEGNA 21/BIS - DOMODOSSOLA
• ELLIOTT COMPUTER SHOP - VIA DON MINZONI 32 - INTRA
• TRISCONI VALERIA - VIA MAZZINI 90 - OMEGNA
Torino
• ABA ELETTRONICA - VIA C. FOSSATI 5/P
• ALEX COMPUTER E GIOCHI - C.SO FRANCIA 333/4
• COMPUTER HOME - VIA SAN DONATO 46/D
• COMPUTING NEW - VIA M. POLO 40/E
• C.D.M. ELETTR. - VIA MAROCHETTI 17
• DE BUG - C.SO V. EMANUELE II 22
• DESME UNIVERSAL - VIA S.SECONDO 95
• FDS ALTERIO - VIA BORGARO 86/D
• IL COMPUTER - VIA N. FABRIZI 126
• MICRONTEL - C.SO D. degli ABRUZZI 28
• PLAY GAMES SHOP - VIA C. ALBERTO 39/E
• RADIO TV MIRAFIORI - C.SO UNIONE SOVIETICA 381
• SMT ELETTRONICA - VIA BIBIANA 83/bis
Provincia di Torino
• PAUL E CHICO VIDEOSOUND - VIA V.EMANUELE 52 - CHIERI
• BIT INFORMATICA - VIA V. EMANUELE 154 - CIRIE'
• HI - FI CLUB - C.SO FRANCIA 920 - COLLEGNO
• MISTER PERSONAL - VIA CATTANEO 52 - FAVRIA
• I.C.S. - VIA TORINO 73 - IVREA
• DAG - VIA I MAGGIO 40 - LUSERNA S. GIOVANNI
• EUREX - C.SO INDIPENDENZA 5 - RIVAROLO CANAVESE
• DIAM INFORMATICA - C.SO FRANCIA 146 bis - RIVOLI
• FULLINFORMATICA - VIA V.VENETO 25 - RIVOLI
• GAMMA COMPUTER - VIA CAVOUR 3A-3B - SET.TORINESE
Vercelli
•ELETTROGAMMA - C.SO BORMIDA 27 ang. V.Montanara
• ELETTRONICA - STRADA TORINO 15
Provincia di Vercelli
• C.S.I. TEOREMA - VIA LOSANA 9 - BIELLA
• SIGEST - VIA BERTODANO 8 - BIELLA
• REMONDINO FRANCO - VIA ROMA 5 - BORGOSESIA
• FOTOSTUDIO TREVISAN - VIA XXV APRILE 24/B - COSSATO
• STUDIO FOTOGRAFICO IMARISIO - P.ZZA M. LIBERTA' 7 - TRINO

VENETO

Belluno
• UP TO DATE - VIA V. VENETO 43
Provincia di Belluno
• GUERRA COMPUTERS - V.LE MAZZINI 10/A - FELTRE
Padova
• BIT SHOP - VIA CAIROLI 11
• COMPUMANIA - VIA T. CAMPOSANPIERO 37
• D.P.R. DE PRATO R. - V.LO LOMBARDO 4
• G.F. MARCATO - VIA MADONNA DELLA SALUTE 51/53
• SARTO COMPUTER - VIA ARMISTIZIO 79
Provincia di Padova
• COMPUTER SERVICE - BORGO TREVISO 150 - CITTADELLA
Treviso
• BIT 2000 - VIA BRANDOLINI D'ADDA 14
• GUERRA EGIDIO & C. - V.LE CAIROLI 95
Provincia di Treviso
• DE MARIN COMPUTERS - VIA MATTEOTTI 142 - CONEGLIANO
• SIDESTREET - VIA SALVO D'ACQUISTO 8 - MONTEBELLUNA
• FALCON ELETTROAUDIOVIDEO - VIA TERRAGGIO 116 - PREGANZIOL
Venezia
• GUERRA EGIDIO & C. - VIA BISSUOLA 20/A - MESTRE
• TELERADIO FUGA - SAN MARCO 3457
Provincia di Venezia
• GUERRA EGIDIO & C. - VIA VIZZOTTO 29 - SAN DONA' DI PIAVE
• REBEL - VIA F. CRISPI 10 - SAN DONA' DI PIAVE
Verona
• CASA DELLA RADIO - VIA CAIROLI 10
• TELESAT - VIA VASCO DE GAMA 8
• Provincia di Verona
• UBER - CP 0363(RAG.SOC. DERTA) - VIA MASCAGNI 31 -CASTE' D'AZZANO
• FERRARIN - VIA DEI MASSARI 10 - LEGNAGO
• COMPUTERS CENTER - VIA CANTORE 26 - VILLAFRANCA
Vicenza
• ELET. BISELLO - V.LE TRIESTE 427/429
• SCALCHI MARKET - VIA CA' BALBI 139
Provincia di Vicenza
• SCHIAVOTTO - VIA ZANELLA 21 - CAVAZZALE
• GUERRA E. & C. - V.LE DELLE INDUSTRIE - MONTECCHIO MAGGIORE

FRIULI VENEZIA GIULIA

Gorizia
• E.CO. ELETTRONICA - VIA F.LLI COSSAR 23
Trieste
• AVANZO GIACOMO - P.ZZA CAVANA 7
• COMPUTER SHOP - VIA P. RETI 6
• COMPUTIGI - VIA XX SETTEMBRE 51
• CTI - VIA PASCOLI 4
Udine
• MOFERT 2 - VIA LEOPARDI 21
• R.T. SISTEM UDINE - VIA L. DA VINCI 99
Provincia di Udine
• IL PUNTO ELETTRONICO - VIA VENDRAMIN 184 - LATISANA
• IDRENO MATTIUSSI &C. - VIA LICINIANA 58 - MARTIGNACCO

TRENTINO ALTO ADIGE

Bolzano
• COMPUTER POINT - VIA ROMA 82/A
• MATTEUCCI PRESTIGE - VIA MUSEO 54
Provincia di Bolzano
• RADIO MAIR-ELECTRO - VIA CENTRALE 70 - BRUNICO
• ELECTRO RADIO HENDRICH - VIA DELLE CORSE 106 - MERANO
• ERICH KONTSCHIEDER - PORTICI 313 - MERANO
• ELECTRO TAPPEINER - P.ZZA PRINCIPALE 90 - SILANDRO
Trento
• CRONST - VIA G. GALILEI 25
Provincia di Trento

• AL RISPARMIO - C.SO VERONA 138 - ROVERETO

LIGURIA
Genova
• ABM COMPUTER - P.ZZA DE FERRARI 24 rosso
• CAPRIOTTI G. - IA MAMIANI 4r - SAMPIERDARENA
• C.tro ELET. - VIA CHIARAVAGNA 10 R - VIA SESTRI 69R
• COM.le SOTTORIPA - VIA SOTTORIPA 115/117
• FOTOMONDIAL - VIA DEL CAMPO 3-5-9-11-13 r
• LA NASCENTE - VIA SAN LUCA 4/1
• PLAY TIME - VIA GRAMSCI 3/5/7 rosso
• RAPPR-EL - VIA BORGORATTI 23 R
Imperia
• CASTELLINO - VIA BELGRANO 44
Provincia di Imperia
• CENTRO HI-FI VIDEO - VIA DELLA REPUBBLICA 38 -SANREMO
• CASTELLINO - VIA GENOVA 48 - VENTIMIGLIA
La Spezia
• I.L. ELETTRONICA - VIA V. VENETO 123
Provincia di La Spezia
• I.L. ELETTRONICA - VIA AURELIA 299 - FORNOLA DI VEZZANO
Savona
• CASTELLINO - C.SO TARDY E BENECH 101
Provincia di Savona
• CELESIA ENZA - VIA GARIBALDI 144 - LOANO

EMILIA
Bologna
• EUROELETTRICA - VIA RANZANI 13/2
• MINNELLA ALTA FEDELTA' - VIA MAZZINI 146/2
• MORINI & FEDERICI - VIA MARCONI 28/C
• STERLINO - VIA MURRI 73/75
Provincia di Bologna
• S.C. COMPUTERS - VIA E. FERMI 4 - CASTEL SAN PIETRO
• S.P.E. INFORMATICA - VIA DI MEZZO PONENTE 385 - CREVALCORE
• ARCHIMEDE SISTEMI - VIA EMILIA 124 - S. LAZZARO DI SAVENA
Modena
• CO - EL - VIA CESARI 7
• ORSA MAGGIORE - P.ZZA MATTEOTTI 20
• VIDEO VAL WILLY COMPUTERS - VIA CANALETTO 223
Provincia di Modena
• NEW MEDIA SYSTEM - VIA ROMA 281 - SOLIERA
Parma
• BABARELLI G. - VIA B. PARENTE 14/A/B
Provincia di Parma
• PONGOLINI - VIA CAVOUR 32 - FIDENZA
Piacenza
• COMPUTER LINE - VIA G. CARDUCCI 4
• DELTA COMPUTER - VIA M. DELLA RESISTENZA 15/G

TEGGIO EMILIA
• COMPUTERLINE - VIA SAN ROCCO 10/C
• POOL SHOP - VIA EMILIA S. STEFANO 9/C
Provincia di Reggio Emilia
• MACCHIONI - VIA STATALE 467 - CASALGRANDE
ROMAGNA
Ferrara
• BUSINESS POINT - VIA CARLO MAYER 85
Forlì
• COMPUTER VIDEO CENTER - VIA CAMPO DI MARTE 122
Provincia di Forlì
• TOP BIT - VIA VENETO 12 - FORLIMPOPOLI
• COMPUTER HOUSE - V.LE TRIPOLI 193/D - RIMINI
• EASY COMPUTER - VIA LAGOMAGGIO 50 - RIMINI

REPUBBLICA S. MARINO
Ravenna
• COMPUTER HOUSE - VIA TRIESTE 134
Provincia di Ravenna
• ARGNANI - P.ZZA DELLA LIBERTA 5/A - FAENZA
• ELECTRON INFORMATICA - VIA F.LLI CORTESI 17 - LUGO
• P.L.Z. INFORMATICA - P.ZZA SERCOGNANI 6 - FAENZA

TOSCANA
Arezzo
• DELTA SYSTEM - VIA PIAVE 13
Firenze
• ATEMA - VIA BENEDETTO MARCELLO 1a-1b
• ELETTRONICA CENTOSTELLE - VIA CENTO STELLE 5/a-b
• HELP COMPUTER - VIA DEGLI ARTISTI 15-A
• TELEINFORMATICA TOSCANA -VIA BRONZINO 36
Provincia di Firenze
• WAR GAMES - VIA R. SANZIO 126/A - EMPOLI
• NEW EVM COMPUTER - VIA DEGLI INNOCENTI 2 - FIGLINE VALDARNO
• C.tro INFOR. - VIA ZNOJMO 41 - PONTASSIEVE
• COSCI F.LLI - VIA ROMA 26 - PRATO
• BARBAGLI C. ELET. - VIA F. BONI 80 - PRATO
Grosseto
• COMPUTER SERVICE - VIA DELL'UNIONE 7
Livorno
• ETA BETA - VIA SAN FRANCESCO 30
• FUTURA 2 - VIA CAMBINI 19
Provincia di Livorno
• PUNTO ROSSO - VIA BARONTINI 28 - PIOMBINO
Provincia di Lucca
• IL COMPUTER - V.LE COLOMBO 216 - LIDO DI CAMAIORE
• SANTI VITTORIO - VIA ROMA 23 - S. ROMANO GARFAGNANA
• TOP GAMES - VIA S. ANDREA 122 - VIAREGGIO
Massa
• EURO COMPUTER - P.ZZA G. BERTAGNINI 4
Carrara
• RADIO LUCONI - VIA ROMA 24/B
Pisa
• ELECTRONIC SERVICE - VIA DELLA VECCHIA TRANVIA 10
• PUCCINI S.- CP 1199 (RAG.SOC. MAREX) - VIA C.CAMMEO 64
• TONY HI-FI - VIA CARDUCCI
Provincia di Pisa
• M.C. INFORMATICA - VIA DEL CHIESINO 4 - PONTEDERA (PI)
Pistoia
• ELECTRONIC SHOP - VIA DEGLI SCALZI 3
Provincia di Pistoia
• ZANNI &C. - C.SO ROMA 45 - MONTECATINI T.

Siena
• R. BROGI - P.ZZA GRAMSCI 28
• VIDEO MOVIE - VIA GARIBALDI 17
Provincia di Siena
• ELETTRONICA di BIFOLCHI - VIA DI GRACCIANO NEL CORSO 111 - MONTEPULCIANO

LAZIO
• CENTRO INF. - D.R.R. srl - TEL. 06-5565672

UMBRIA
Perugia
• MIGLIORATI - VIA S. ERCOLANO 3-10
Provincia di Perugia
• COMPUTER STUDIO'S - VIA IV NOVEMBRE 18/A - BASTIA UMBRA
• WARE - VIA DEI CASCERI 31 - CITTA' DI CASTELLO
Terni
• CGS SOFTWARE HOUSE - VIA DONIZETTI 71/A

BASILICATA
Matera
• G. GAUDIANO ELECTRONICS - VIA ROMA ang. XX SETTEMBRE 1

PUGLIA
Bari
• ARTEL - VIA GUIDO D'ORSO 9
• COMPUTER'S ARTS - V.LE MEUCCI 12/B
• PAULICELLI S. & F. - VIA FANELLI 231/C
Provincia di Bari
• F. FAGGELLA - C.SO GARIBALDI 15 - BARLETTA
• G.FAGGELLA - P.ZZA D'ARAGONA 62A - BARLETTA
• LONUZZO G. - VIA NIZZA 21 - CASTELLANA
• TECNOUFF. - VIA RICASOLI 54 - MONOPOLI
• TANGORRA N. - C.SO V.EMANUELE 130/B - TRIGGIANO
Brindisi
• MARANGI E NICCOLI - VIA PROV. SAN VITO 165
Provincia di Brindisi
• MILONE G. - VIA S.F. D'ASSISI 219 - FRANCAVILLA FONTANA
Foggia
• BOTTICELLI G. - VIA SAV POLLICE 2
• E.C.I. COMPUTER - VIA ISONZO 28
• LA TORRE - V.LE MICHELANGELO 185
Provincia di Foggia
• IL DISCOBOLO - VIA T. SOLIS 15 - SAN SEVERO
Lecce
• BIT - VIA 95 REGG.NTO FANTERIA 87/89
Provincia di Lecce
• TECNO UFFICIO - P.ZZA GIOVANNI XXIII 10 - GALLIPOLI
• CEDOK INFORMATICA - VIA UMBERTO I 116 - TRICASE
Taranto
• ELETTROJOLLY C.tro - VIA DE CESARE 13
• TEA - TEC. ELET. AV. - VIA R. ELENA 101

CAMPANIA
Provincia di Avellino
• FLIP FLOP - VIA APPIA 68 - ATRIPALDA
Benevento
• E.CO. INF. - VIA PEPICELLI 21/25
Caserta
• ENTRY POINT - VIA COLOMBO 31
• O.P.C. - VIA G. M. BOSCO 24
Provincia di Caserta
• M.P. COMPUTER - VIA NAPOLI 30 - MADDALONI
• DAMIANO - C.SO V. EMANUELE 23 -ORTA DI ATELLA
• FUSCO B. - VIA NAPOLI 24 - VAIRANO PATERNORA (FRAZ. VAIRANO SCALO)
• LINEA CONTABILE - VIA OSPEDALE 72/76 - SESSA A. (CE)
Napoli
• BABY TOYS - VIA CISTERNA DELL'OLIO 5/BIS
• CASA MUSICALE RUGGIERO - P.ZZA GARIBALDI 74 (INT. STAZ. F.F. S.S.)
• C.tro ELET. CAMPANO - VIA EPOMEO 121
• CI.AN - GALLERIA VANVITELLI 32
• CINE NAPOLI - VIA S. LUCIA 93/95
• DARVIN - CALATA SAN MARCO 26
• GIANCAR 2 - P.ZZA GARIBALDI 37
• ODORINO - L.GO LALA 22 A-B
• R 2 - VIA F. CILEA 285
• SAGMAR - VIA S. LUCIA 140
• TOP VIDEO - TOP COMPUTER - VIA S. ANNA DEI LOMBARDI 12
• VIDEOFOTOMARKET - VIA S. BRIGIDA 19
Provincia di Napoli
• ELECTRONIC DAY - VIA DELLE PUGLIE 17 - CASORIA
• TUFANO - S.S. SANNITICA 87 KM 7 - CASORIA
• SOF SUD - V.LE EUROPA 59 - CASTEL/MARE DI STABIA
• ELETTRONICA 2000 - C.SO DURANTE 40 - FRATTAMAGGIORE
• SPADARO - VIA ROMANI 93 - MADONNA DELL'ARCO
• GATEWAY - VIA NAPOLI 68 - MUGNANO
• VISPINI & DI VUOLO - VIA A.ROSSI 4 - POMPEI
• SPY CASH & CARRY - P.ZZA ARENELLA 6/A - NAPOLI
• NUOVA INFORMATICA SHOP - VIA LIBERTA' 185/191 PORTICI
• BASIC COMPUTER - C.SO GARIBALDI 34 - POZZUOLI
• V.C. - C.SO SECONDIGLIANO 562/B - SECONDIGLIANO
• F. ELETTRONICA - VIA SARNO 102 - STRIANO
• TECNO - VIA V. VENETO 48 - TORRE DEL GRECO
Salerno
• COMPUMARKET - VIA BELVEDERE 35
• COMPUTER MARKET - C.SO VITTORIO EMANUELE 23
Provincia di Salerno
• KING COMPUTER - VIA OLEVANO 56 - BATTIPAGLIA
• DIMER POINT - V.LE AMENDOLA 36 - EBOLI
• IACUZIO F. - VIA MUNICIPIO 14 - MERCATO SAN SEVERINO
• COMPUTER SERVICE - VIA L.DA VINCI 81 - SCAFATI

CALABRIA
Catanzaro
• C. & G. COMPUTER - VIA F. ACRI 28
• PAONE S. & F. - VIA F. ACRI 93/99
Provincia di Catanzaro
• COMPUTER HOUSE - VIA BOLOGNA (L.GO OSPEDALE) - CROTONE
• RIOLO F.LLI - VIA VENEZIA 1/7 - CROTONE
• ING. FUSTO S. - C.SO NICOTERA 99 - LAMEZIA TERME
Cosenza
• MAISON DE L'INFORMATIQUE - VIA PASQUALE ROSSI 34/C
• SIRANGELO COMP. - VIA N. PARISIO 25
Provincia di Cosenza
• HI-FI ALFANO G. - VIA BALDACCHINI 109 - AMANTIA
• ELIGIO ANNICCHIARICO &C. - VIA ROMA 21 - CASTROVILLARI
• ALFA COMPUTER - VIA NAZIONALE 341/A - CORIGLIANO SCALO
REGGIO CALABRIA
• CONTROL SYSTEM - VIA S.F DA PAOLA 49 D
• SYSTEM HOU. - VIA FIUME ang. PALESTINO 1
Provincia di Reggio Calabria
• COMPUTER SHOP - V.LE MATTEOTTI 36/38 - LOCRI
• PICIEFFE - C.SO F. S. ALESSIO 19 - TAURIANOVA
SICILIA
• CENTRO INF. - ITALSOFT SRL - TEL. 0935-696090

Leggo VR perché mi dà la rotta